Package: admixtools (via r-universe)

June 2, 2026

Type Package

Title Inferring demographic history from genetic data

Version 2.0.10

Description admixtools is a set of methods used to infer demographic history from genetic data using f-statistics. It was originally implemented as a command line program in C. This is a new implementation of the programs qpDstat, qpAdm, qpGraph, and contains several new features. It is efficient because it precomputes f2-statistics which can be re-used for many analyses.

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

LazyData true

SystemRequirements C++17

URL https://github.com/uqrmaie1/admixtools

BugReports https://github.com/uqrmaie1/admixtools/issues

LinkingTo Rcpp, RcppArmadillo

Imports abind, cli, crayon, cubelyr, data.tree, digest, doParallel, dplyr (>= 1.1.0), foreach, furrr, ggplot2, grDevices, igraph (>= 2.1.0), jsonlite, lobstr, magrittr, pgenlibr, quadprog, purrr, Rcpp, readr, rlang, stringi, stringr, tibble (>= 3.1.0), tidyr (>= 1.0.0), utils

Suggests admixturegraph, DT, future, genio, ggforce, gridExtra, htmlwidgets, knitr, m2r, plotly, pracma, pROC, Rgraphviz, rmarkdown, shiny, shinyjs, shinyBS, shinyFiles, shinyalert, shinythemes, shinyWidgets, shinydashboard, testthat (>= 3.0.0), tidyverse, withr

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Config/roxygen2/version 8.0.0
Config/testthat/edition 3
Config/pak/sysreqs libglpk-dev libicu-dev libxml2-dev libx11-dev
Repository https://evolecolgroup.r-universe.dev
Date/Publication 2026-06-02 15:59:17 UTC
RemoteUrl https://github.com/uqrmaie1/admixtools
RemoteRef master
RemoteSha 0c3ce6a408888bb2241ce4a08c6f06e45e774aa0

Contents

`add_sampled_tips` *Materialise sampled internal nodes as leaf tips.*

Description

qpgraph models every fitted population as a graph tip. A population marked `samples` in the nodes tibble that sits at an internal (ancestral) position cannot be fit directly: forcing it into the population set collapses the edge to its descendants to ~0 and the score blows up by orders of magnitude. This transform rewrites each sampled internal node `X` into a new unlabelled ancestor `X<suffix>` (inheriting X's parents and children) plus `X` as a leaf tip off that ancestor, the standard tip/sibling representation. After the transform `X` is a topological leaf, so the existing leaf-based fitting path uses it with no further change. A sampled internal node that is itself an admixture node (in-degree 2) keeps both parent edges on the new ancestor, so the admixture is preserved.

Usage

```
add_sampled_tips(graph, suffix = "_anc")
```

Arguments

`graph`	An edge tibble carrying a `nodes` attribute (e.g. from `read_lgo`). Graphs with no sampled internal nodes are returned unchanged.
`suffix`	Suffix for the new ancestor node names (default `"_anc"`).

Value

An edge tibble in tip/sibling form, fittable by `qpgraph`. Every edge column of the input is carried through; the new ancestor->tip edges take `NA` for fitted quantities (e.g. `weight`, `time`) and the input's non-admix `type` literal. The input's `nodes` attribute is dropped, because it is keyed to the old structure and the formerly-internal sampled nodes are now leaves.

See Also

`set_node_attrs`

---

admixtools *Tools for inferring demographic history from genetic data*

Description

admixtools is a collection of several programs for calculating f-statistics and using them to make inferences about demographic history from genetic data Aside from the core functions `qp3pop`, `qpdstat`, `qpadm`, and `qpgraph`, there are also wrapper functions and parsing functions around the original ADMIXTOOLS software, as well as functions to read genotype files and precompute the data necessary to quickly compute f statistics.

Author(s)

Robert Maier `<robertmaier@gmx.net>`

Nick Patterson

References

Patterson, N. et al. (2012) *Ancient admixture in human history.* Genetics

See Also

`https://uqrmaie1.github.io/admixtools/index.html`

---

afs_to_counts *Compute count blocks and write them to disk*

Description

This is intended for computing allele count data for a large number of individuals, too many to do everything in working memory. It assumes that the allele frequencies have already been computed and are stored in `.rds` files, split into consecutive blocks for a set of individuals.

Usage

```
afs_to_counts(
  genodir,
  outdir,
  chunk1,
  chunk2,
  overwrite = FALSE,
  verbose = TRUE
)
```

Arguments

genodir	Directory with genotype files split in chunks created by extract_afs (with each individual its own population).
outdir	Directory where allele count data will be stored
chunk1	Index of the first chunk of individuals
chunk2	Index of the second chunk of individuals
overwrite	Overwrite existing files (default FALSE)
verbose	Print progress updates

See Also

extract_counts Does the same thing in one step for smaller data.

---

afs_to_f2 *Compute f2 blocks and write them to disk*

Description

This is intended for computing f2-statistics for a large number of populations, too many to do everything in working memory. It assumes that the allele frequencies have already been computed and are stored in .rds files, split into consecutive blocks for a set of populations. This function calls write_f2, which takes a (sub-)chunk of pairwise f2-statistics, and writes one pair at a time to disk.

Usage

```
afs_to_f2(
  afdir,
  outdir,
  chunk1,
  chunk2,
  blgsize = 0.05,
  snpwt = NULL,
  overwrite = FALSE,
  type = "f2",
```

```
  poly_only = FALSE,
  snpdat = NULL,
  apply_corr = TRUE,
  verbose = TRUE
)
```

Arguments

afdir	Directory with allele frequency and counts `.rds` files created by split_mat
outdir	Directory where data will be stored
chunk1	Index of the first chunk of populations
chunk2	Index of the second chunk of populations
blgsize	SNP block size in Morgan. Default is 0.05 (5 cM). If `blgsize` is 100 or greater, if will be interpreted as base pair distance rather than centimorgan distance.
snpwt	A vector of scaling factors applied to the f2-statistics for each SNP. The length has to match the number of SNPs.
overwrite	Overwrite existing files (default `FALSE`)
type	Type of statistic to compute: `'f2'` (default), `'ap'` (allele frequency products), or `'fst'`.
poly_only	Only use polymorphic SNPs (default `FALSE`).
snpdat	SNP metadata data frame; if `NULL`, loaded from `snpdat.tsv.gz` in `afdir`.
apply_corr	Apply bias correction to f2 estimates (default `TRUE`).
verbose	Print progress updates

See Also

extract_f2 Does the same thing in one step for smaller data.

Examples

```
## Not run:
afdir = 'tmp_af_dir/'
f2dir = 'f2_dir'
extract_afs('path/to/packedancestrymap_prefix', afdir)
numchunks = length(list.files(afdir, 'afs.+rds'))
# numchunks should be the number of split allele frequency files
for(i in 1:numchunks) {
  for(j in i:numchunks) {
    afs_to_f2(afdir, f2dir, chunk1 = i, chunk2 = j)
  }
}

## End(Not run)
# Alternatively, the following code will do the same, while submitting each chunk as a separate job.
# (if \code{\link[future]{plan}} has been set up appropriately)
## Not run:
```

```
furrr::future_map(1:numchunks, ~{i=.; map(i:numchunks, ~{
  afs_to_f2(afdir, f2dir, chunk1 = i, chunk2 = .)
  })})

## End(Not run)
```

`afs_to_f2_blocks` *Compute all pairwise f2 statistics*

Description

This function takes a allele frequency data and computes blocked f2 statistics for all population pairs, which are written to `outdir`. $f2$ for each SNP is computed as $(p1-p2)^2 - p1(1-p1)/(n1-1) - p2(1-p2)/(n2-1)$, where $p1$ and $p2$ are allele frequencies in populations 1 and 2, and $n1$ and $n2$ is the number of non-missing haplotypes in populations 1 and 2. See `details`

Usage

```
afs_to_f2_blocks(
  afdat,
  maxmem = 8000,
  blgsize = 0.05,
  pops1 = NULL,
  pops2 = NULL,
  outpop = NULL,
  outdir = NULL,
  overwrite = FALSE,
  afprod = TRUE,
  fst = TRUE,
  poly_only = c("f2"),
  apply_corr = apply_corr,
  n_cores = 1,
  verbose = TRUE
)
```

Arguments

`afdat`	A list with three items with the same SNP in each row (generated by `packedancestrymap_to_afs` or `plink_to_afs`) • `afs` A matrix of allele frequencies for all populations (columns) and SNPs (rows) • `counts` A matrix of allele counts for all populations (columns) and SNPs (rows) • `snpdat` A data frame with SNP metadata
`maxmem`	split up allele frequency data into blocks, if memory requirements exceed `maxmem` MB.

`blgsize`	SNP block size in Morgan. Default is 0.05 (5 cM). If `blgsize` is 100 or greater, if will be interpreted as base pair distance rather than centimorgan distance.
`pops1`	Populations for the first dimension; if `NULL` (default), all populations are used.
`pops2`	Populations for the second dimension; if `NULL` (default), all populations are used.
`outpop`	If specified, f2-statistics will be weighted by heterozygosity in this population
`outdir`	Directory into which to write f2 data (if `NULL`, data is returned instead)
`overwrite`	Should existing files be overwritten? Only relevant if `outdir` is not `NULL`
`afprod`	Compute allele frequency products in addition to f2-statistics (default `TRUE`).
`fst`	Compute FST in addition to f2-statistics (default `TRUE`).
`poly_only`	Exclude sites with identical allele frequencies in all populations. Can be different for f2-statistics, allele frequency products, and fst. Should be a character vector of length three, with some subset of `c("f2", "ap", "fst")`
`apply_corr`	Apply bias correction to f2 estimates (default `TRUE`).
`n_cores`	Number of cores for parallel computation (default 1).
`verbose`	Print progress updates

Details

For each population pair, each of the $i = 1, \ldots, n$ resutling values (n is around 700 in practice) is the mean $f2$ estimate across all SNPs except the ones in block i.

$-p1(1-p1)/(2n1-1) - p2(1-p2)/(2n2-1)$ is a correction term which makes the estimates unbiased at low sample sizes.

Examples

```
## Not run:
afdat = plink_to_afs('/my/geno/prefix')
afs_to_f2_blocks(afdat, outdir = '/my/f2/data/')

## End(Not run)
```

`agraph_to_igraph` *Convert agraph to igraph*

Description

`agraph` is the format used by the `admixturegraph` packge. `igraph` is used by the `admixtools` package

Usage

```
agraph_to_igraph(agraph)
```

Arguments

agraph	An admixture graph in agraph format

Value

An admixture graph in igraph format

Examples

```
## Not run:
agraph_to_igraph(agraph)

## End(Not run)
```

---

as_edge_tibble *Coerce to a plain edge tibble (drop the nodes attribute).*

Description

Use this to opt out of the nodes-aware API and revert to the legacy edge-only representation. Warns if the dropped nodes tibble carried any non-NA data.

Usage

```
as_edge_tibble(graph)
```

Arguments

graph	An edge tibble (potentially with a nodes attribute).

Value

The edge tibble with attr(., "nodes") cleared.

boo_list — *Generate a list of bootstrap resampled arrays*

Description

Generate a list of bootstrap resampled arrays

Usage

```
boo_list(arr, nboot = dim(arr)[3])
```

Arguments

arr	3d array with blocked estimates, with blocks in the 3rd dimension
nboot	Number of bootstrap iterations

Value

A list of bootstrap resampled arrays, with 3rd dimension equal to `nboot`

See Also

loo_list est_to_boo qpgraph_resample_snps2

compare_fits — *Compare the fit of two qpgraph models*

Description

Takes the bootstrap score distribution of two fits on the same populations and tests whether the scores of one graph are significantly higher or lower than the scores of the other graph.

Usage

```
compare_fits(scores1, scores2)
```

Arguments

scores1	Scores for the first graph. Each score should be for same model evaluated on different bootstrap samples of the SNP blocks. See qpgraph_resample_multi
scores2	Scores for the second graph, evaluated on the same bootstrap samples as the first graph

Value

A list with statistics comparing the two models

- `p_emp`: The two-sided bootstrap p-value testing whether the scores of one model are higher or lower than the scores of the other model. It is two times the fraction of bootstrap replicates in which model 1 has a lower score than model 2 (or vice-vera, whichever is lower). This is simply a bootstrap test for testing whether some quantity (the the score difference of two models in this case) is significantly different from zero.
- `p_emp_nocorr`: `p_emp` is truncated to be no less than the reciprocal of the number of bootstrap iterations (the length of the score vectors). `p_emp_nocorr` is not truncated and can be equal to 0.
- `ci_low`: The 2.5% quantile of distribution of score differences
- `ci_high`: The 97.5% quantile of distribution of score differences
- The other items in this list are less important and can be ignored. In particular, `p` is not as reliable as `p_emp` because it assumes that the score differences follow a normal distribution.

See Also

qpgraph_resample_multi

Examples

```
## Not run:
fits = qpgraph_resample_multi(f2_blocks, list(graph1, graph2), nboot = 100)
compare_fits(fits[[1]]$score, fits[[2]]$score)

## End(Not run)
```

---

compare_fits2 *Compare the fit of two qpgraph models*

Description

Takes two data frames with model fits computed on two graphs for on the same populations and tests whether the scores of one graph are significantly better than the scores of the other

Usage

```
compare_fits2(fits1, fits2, boot = FALSE)
```

Arguments

`fits1`	The fits of the first graph
`fits2`	The fits of the second graph
`boot`	should match the `boot` parameter in `qpgraph_resample_snps` (`FALSE` by default)

Examples

```
## Not run:
nblocks = dim(example_f2_blocks)[3]
train = sample(1:nblocks, round(nblocks/2))
fits1 = qpgraph_resample_snps(example_f2_blocks[,,train], graph = graph1,
                              f2_blocks_test = example_f2_blocks[,,-train])
fits2 = qpgraph_resample_snps(example_f2_blocks[,,train], graph = graph2,
                              f2_blocks_test = example_f2_blocks[,,-train])
compare_fits2(fit1, fit2)

## End(Not run)
```

---

compare_fits4 *Compare the fit of two qpgraph models*

Description

Takes two data frames with model fits computed on two graphs for on the same populations and tests whether the scores of one graph are significantly better than the scores of the other.

Usage

```
compare_fits4(fit1, fit2, f2_blocks, f2_blocks_test, boot = FALSE, seed = NULL)
```

Arguments

`fit1`	The fit of the first graph
`fit2`	The fit of the second graph
`f2_blocks`	f2 blocks used for fitting `fit1` and `fit2`. Used in combination with `f2_blocks_test` to compute f-statistics covariance matrix.
`f2_blocks_test`	f2 blocks which were not used for fitting `fit1` and `fit2`
`boot`	If `FALSE` (the default), block-jackknife resampling will be used to compute standard errors. Otherwise, block-bootstrap resampling will be used to compute standard errors. If `boot` is an integer, that number will specify the number of bootstrap resamplings. If `boot = TRUE`, the number of bootstrap resamplings will be equal to the number of SNP blocks. If bootstrap resampling is enabled, empirical p-values (`p_emp`) and 95 confidence intervals (`ci_low` and `ci_high`) will be reported.
`seed`	Random seed used if `boot` is `TRUE`. Does not need to match a seed used in fitting the models

Examples

```
## Not run:
nblocks = dim(example_f2_blocks)[3]
train = sample(1:nblocks, round(nblocks/2))
fit1 = qpgraph(example_f2_blocks[,,train], graph1)
fit2 = qpgraph(example_f2_blocks[,,train], graph2)
compare_fits4(fit1, fit2, example_f2_blocks[,,train], example_f2_blocks[,,-train])

## End(Not run)
```

---

compute_f2_cache_id *Compute a stable cache identifier for an* extract_f2 *run*

Description

Returns a string of the form "sha256:<hex>" that uniquely identifies the inputs that determined the f2 blocks produced by extract_f2(). Two runs over byte-equivalent inputs (same indfile, same variants, same arguments, same schema version) produce byte-equal hashes; any change to any contributing input changes the hash.

Usage

```
compute_f2_cache_id(
  pref,
  format = NULL,
  inds = NULL,
  pops = NULL,
  snpfile_kept = NULL,
  blgsize = 0.05,
  extra_args = list()
)
```

Arguments

pref	Either a genotype prefix (same as passed to extract_f2(), any format extract_f2() accepts) or an existing f2-output directory (the value passed to extract_f2() as outdir).
format	Genotype format hint when pref is a prefix. One of 'plink', 'packedancestrymap', 'eigenstrat', 'pfile', or NULL to detect from file existence at pref. Ignored when pref is a directory.
inds, pops	Individual / population filters (same as passed to extract_f2()). Ignored when pref is a directory.
snpfile_kept	A data.frame with at least CHR, POS, A1, A2 columns, representing the SNPs that survived filtering. Required when computing fresh from a genotype prefix; ignored when pref is a directory.
blgsize	Block size in Morgans (same as passed to extract_f2()). Ignored when pref is a directory.

`extra_args` Named list of additional `extract_f2` arguments whose values affect output. Captured verbatim into the hash payload. **Values must be primitives** (scalars, atomic vectors, or nested lists thereof): non-primitives like environments, closures, or external pointers are serialized via R's session-internal `serialize()`, which produces session-dependent bytes and breaks the determinism guarantee. Ignored when `pref` is a directory.

Details

When `extract_f2()` is given an `outdir`, the cache id is also written as a single line to `<outdir>/.f2_cache_id`. Orchestrators can read this file to decide whether previously cached downstream artefacts (qpAdm / qpGraph results computed against these blocks) are still current, without having to hash hundreds of MB of `.rds` block files or rely on mtimes.

Two input modes:

- **Genotype prefix** (typical: invoked internally by `extract_f2()`): recomputes the hash from the on-disk inputs and filter args. Requires `snpfile_kept` (the post-filter SNP table) because variant filtering isn't reproducible from the prefix alone.
- **f2 directory** (typical: orchestrator probe of an existing cache): reads the `.f2_cache_id` sidecar previously written by `extract_f2()`. Cheap; no genotype-file access. Most callers want this mode. If the sidecar is absent (e.g. extract_f2 was killed after writing `cache_metadata.json` but before the sidecar), the call falls back to reading `cache_id` from `cache_metadata.json` in the same directory.

The hash payload (when computing fresh) is:

- **Sample set**: sorted list of IIDs that contributed to the f2 blocks, resolved from the input `.fam` / `.ind` / `.psam` according to `inds` and `pops`.
- **Variant set**: sorted `(CHR, POS, A1, A2)` tuples of the SNPs that survived all filtering (`auto_only`, `transitions`, `transversions`, `keepsnps`, `maxmiss`, `minmaf`, `maxmaf`, `minac2`).
- **Block boundaries**: per-block lengths produced by `get_block_lengths()` with the supplied `blgsize`.
- **Schema version**: a package-private constant (`.f2_cache_id_schema`) that the maintainer bumps when extract_f2's hash-affecting logic changes. Decoupled from `packageVersion()` so pure bug-fix releases don't invalidate every cache on disk.
- **Filter arguments**: the additional `extract_f2` arguments that affect which SNPs are kept and how blocks are formed.

Value

A character scalar of the form `"sha256:<64 hex chars>"`.

See Also

`extract_f2()`

count_snps *Count SNPs in an f2-statistics array*

Description

This function adds up all block lengths (number of SNPs in each SNP blocks), which are stored in the names along the third dimension of the array. When the f2-statistics were computed while setting `maxmiss` to values greater than 0, it is possible that not all f2-statistics are based on the same number of SNPs. In that case, the value returned by this function is merely an upper bound on the number of SNPs used.

Usage

```
count_snps(f2_blocks)
```

Arguments

`f2_blocks`	A 3d array of per block f2-statistics

Value

The total number of SNPs across all blocks

See Also

`f2_from_geno`, `f2_from_precomp`

count_zero_edges *Count zero-length edges*

Description

`agraph` is the format used by the `admixturegraph` packge. `igraph` is used by the `admixtools` package

Usage

```
count_zero_edges(edges, epsilon = 1e-06)
```

Arguments

`edges`	Edges data frame from fitted admixture graph
`epsilon`	Every edge with length

Value

The number of edges with length $<$ epsilon

Examples

```
## Not run:
fit = qpgraph(example_f2_blocks, example_igraph)
count_zero_edges(fit$edges)

## End(Not run)
```

`decomposed_tree_neighbors` *Find all trees within SPR distance of 1 of all graph component trees*

Description

Returns all trees which can be reached through one iteration of subtree-prune-and-regraft on any graph component tree

Usage

```
decomposed_tree_neighbors(graph)
```

Arguments

`graph`	An admixture graph

Value

A data frame with all trees

`default_drift_to_time` *Default drift-to-time conversion (identity)*

Description

The default `drift_to_time` function for `graph_to_lgo()`. Returns `drift_vec` unchanged (treats drift values directly as LEGOFIT time starting values).

Usage

```
default_drift_to_time(drift_vec, segment_vec = NULL)
```

Arguments

`drift_vec`	numeric vector of per-edge drift values.
`segment_vec`	character vector of segment names (the `to` endpoint of each edge); ignored by this default but required by the contract so user-supplied alternatives can index per-segment.

Details

Supply a custom function here when you want to convert drift values to absolute generation times before writing the `.lgo`. The function must accept two arguments: `drift_vec` (numeric) and `segment_vec` (character), and return a numeric vector of the same length as `drift_vec`.

Value

Numeric vector, same length as `drift_vec`.

`delete_admix` *Delete an admixture edge*

Description

Delete an admixture edge

Usage

```
delete_admix(graph, from = NULL, to = NULL)
```

Arguments

`graph`	An admixture graph
`from`	Edge source node
`to`	Edge target node

Value

Admixture graph with one deleted edge

See Also

`insert_admix`

delete_groups *Delete groups*

Description

This function deletes data for groups created by `group_samples`

Usage

```
delete_groups(dir, groups = NULL, verbose = TRUE)
```

Arguments

`dir`	Directory with precomputed individual pair data
`groups`	Groups to delete. Defaults to all groups
`verbose`	Print progress updates

Value

Invisibly returns sample IDs in deleted groups as character vector

See Also

`group_samples`

Examples

```
## Not run:
dir = 'my/f2/dir/'
inds = c('ind1', 'ind2', 'ind3', 'ind4', 'ind5')
pops = c('pop_A', 'pop_A', 'pop_A', 'pop_B', 'pop_B')
group_samples(dir, inds, pops)

## End(Not run)
```

delete_leaf *Remove population from graph*

Description

Remove population from graph

Usage

```
delete_leaf(graph, leaf)
```

Arguments

graph	An admixture graph
leaf	Population to be removed

Value

Admixture graph with removed population

See Also

insert_leaf

---

desimplify_graph *Add two nodes before each admixture node*

Description

This is used to revert simplify_graph.

Usage

```
desimplify_graph(graph)
```

Arguments

graph	An admixture graph

Examples

```
simple = simplify_graph(example_igraph)
desimple = desimplify_graph(simple)
plot_graph(simple)
plot_graph(desimple)
```

---

discard_from_aftable *Filter SNPs in an allele-frequency table*

Description

Applies SNP-level filters to an allele-frequency object (the kind returned by plink_to_afs(), eigenstrat_to_afs(), or packedancestrymap_to_afs()) and returns the same shape with the dropped SNPs removed. This is the filter that extract_f2() applies internally between reading allele frequencies from disk and computing per-pair f2 blocks; it is exported here so callers who assemble the AFS-direct pipeline manually can apply the same filter without reaching into the package via :::.

Usage

```
discard_from_aftable(
  afdat,
  maxmiss = 0,
  minmaf = 0,
  maxmaf = 0.5,
  minac2 = FALSE,
  outpop = NULL,
  auto_only = TRUE,
  poly_only = FALSE,
  transitions = TRUE,
  transversions = TRUE,
  keepsnps = NULL
)
```

Arguments

`afdat`	An allele-frequency table: a list with elements `afs` (nsnp x npop matrix of reference-allele frequencies), `counts` (nsnp x npop matrix of observed allele counts), and `snpfile` (a tibble of SNP metadata with at least `SNP`, `CHR`, `A1`, `A2`).
`maxmiss`	Drop SNPs where the fraction of populations with zero called alleles exceeds this value. Default `0`: drop any SNP missing from at least one population. Set to `1` to disable.
`minmaf`	Minimum minor-allele frequency (computed as a count-weighted row mean across populations). Default `0` (no filter).
`maxmaf`	Maximum minor-allele frequency. Default `0.5`.
`minac2`	If `TRUE` (or `2`), drop SNPs whose minimum per-population allele count is below 2. With `minac2 = 2`, only non-singleton populations are considered when computing the minimum. Default `FALSE`.
`outpop`	Optional name of an outgroup population. When supplied, adds an `outgroupaf` column to the SNP metadata equal to the allele frequency of `outpop` at each SNP; otherwise the column is set to a sentinel value of `0.5` (the midpoint between 0 and 1), which makes any subsequent outgroup-based filter a no-op.
`auto_only`	Drop SNPs on non-autosomal chromosomes (anything outside `1:22`). Default `TRUE`.
`poly_only`	Drop SNPs that are monomorphic across the included populations. Default `FALSE`.
`transitions`	Keep transition SNPs (A/G and C/T). Default `TRUE`. Set to `FALSE` for ancient-DNA studies that want to exclude transitions to avoid post-mortem C->T / G->A deamination artifacts.
`transversions`	Keep transversion SNPs. Default `TRUE`.
`keepsnps`	Optional character vector of SNP IDs to retain. When supplied, overrides **all other filters** (`maxmiss`, `auto_only`, `poly_only`, `minmaf`, `maxmaf`,

minac2, transitions, transversions) – only the SNPs whose IDs appear in keepsnps are kept, regardless of any other filter setting. Default NULL.

Value

A list with the same names as afdat (afs, counts, snpfile), restricted to SNPs that pass all active filters. Errors if zero SNPs remain.

Examples

```
## Not run:
# Read allele frequencies for two populations from a PLINK1 dataset,
# then drop any SNP that is missing in at least one population and any
# non-autosomal site:
afdat = plink_to_afs("/path/to/prefix", pops = c("PopA", "PopB"))
afdat_filt = discard_from_aftable(afdat, maxmiss = 0, auto_only = TRUE)
nrow(afdat_filt$afs) <= nrow(afdat$afs)   # TRUE

## End(Not run)
```

---

edges_to_igraph *Convert data frame graph to igraph*

Description

Convert data frame graph to igraph

Usage

```
edges_to_igraph(edges)
```

Arguments

edges	An admixture graph as an edge list data frame

Value

An igraph object

`eigenstrat_to_afs` *Read allele frequencies from* EIGENSTRAT *files*

Description

Read allele frequencies from *EIGENSTRAT* files

Usage

```
eigenstrat_to_afs(
  pref,
  inds = NULL,
  pops = NULL,
  numparts = 100,
  adjust_pseudohaploid = TRUE,
  verbose = TRUE
)
```

Arguments

`pref`	Prefix of *EIGENSTRAT* files (files have to end in `.geno`, `.ind`, `.snp`)
`inds`	Individuals from which to compute allele frequencies
`pops`	Populations from which to compute allele frequencies. If `NULL` (default), populations will be extracted from the third column in the `.ind` file. If population labels are provided, they should have the same length as `inds`, and will be matched to them by position
`numparts`	Number of parts into which the genotype file is split. Lowering this number can speed things up, but will take more memory.
`adjust_pseudohaploid`	Genotypes of pseudohaploid samples are usually coded as `0` or `2`, even though only one allele is observed. `adjust_pseudohaploid` ensures that the observed allele count increases only by `1` for each pseudohaploid sample. If `TRUE` (default), samples that don't have any genotypes coded as `1` among the first 1000 SNPs are automatically identified as pseudohaploid. This leads to slightly more accurate estimates of f-statistics. Setting this parameter to `FALSE` is equivalent to the ADMIXTOOLS `inbreed: NO` option. Setting `adjust_pseudohaploid` to an integer `n` will check the first `n` SNPs instead of the first 1000 SNPs.
`verbose`	Print progress updates

Value

A list with three data frames: allele frequency data, allele counts, and SNP metadata

Examples

```
## Not run:
afdat = eigenstrat_to_afs(prefix, pops = pops)
afs = afdat$afs
counts = afdat$counts

## End(Not run)
```

`est_to_boo` *Turn per-block estimates into bootstrap estimates*

Description

This works for any statistics which, when computed across `N` blocks, are equal to the weighted mean of the statistics across the `N` blocks.

Usage

```
est_to_boo(arr, nboot = dim(arr)[3], block_lengths = NULL)
```

Arguments

`arr`	3d array with blocked estimates, with blocks in the 3rd dimension.
`nboot`	Number of bootstrap iterations
`block_lengths`	Optional vector of block lengths; if `NULL`, parsed from the 3rd-dimension names of `arr`.

Value

A 3d array with bootstrap estimates. The first two dimensions are equal to those of `arr`. The 3rd dimension is equal to `nboot`.

See Also

`est_to_loo`

est_to_loo *Turn per-block estimates into leave-one-out estimates*

Description

This works for any statistics which, when computed across `N` blocks, are equal to the weighted mean of the statistics across the `N` blocks.

Usage

```
est_to_loo(arr, block_lengths = NULL)
```

Arguments

`arr`	3d array with blocked estimates, with blocks in the 3rd dimension
`block_lengths`	Optional block lengths. If `NULL`, will be parsed from 3rd dimnames in blocks

Value

A 3d array with leave-one-out estimates for jackknife. Dimensions are equal to those of `arr`.

See Also

`loo_to_est` `est_to_boo`

example_anno *Data frame with sample annotations*

Description

Data frame with sample annotations

Usage

```
example_anno
```

Format

A data frame with sample annotations

`example_f2_blocks` *Blocked f2-statistics for 7 populations*

Description

Blocked f2-statistics for 7 populations

Usage

```
example_f2_blocks
```

Format

A 3d array with populations along the first two dimensions, and SNP blocks along the 3rd dimension

---

`example_f2sim1` *Simulated f2-statistics for 5 populations*

Description

Simulated f2-statistics for 5 populations

Usage

```
example_f2sim1
```

Format

A 3d array of f2-statistics computed from msprime simulation output, with populations along the first two dimensions and SNP blocks along the 3rd

---

`example_graph` *Admixture graph for 7 populations*

Description

Admixture graph for 7 populations

Usage

```
example_graph
```

Format

A two column matrix where each row represents one edge

`example_igraph` *Admixture graph for 7 populations*

Description

Admixture graph for 7 populations

Usage

```
example_igraph
```

Format

An `igraph` object

`example_opt` *Data frame with one fitted admixture graph*

Description

Data frame with one fitted admixture graph

Usage

```
example_opt
```

Format

A data frame with a fitted admixture graph, generated by `find_graphs_old`

`example_qpgraph_ref_results` *example_graph fitted using qpGraph*

Description

example_graph fitted using qpGraph

Usage

```
example_qpgraph_ref_results
```

Format

A list with parsed qpGraph output

example_triples *Data frame with population triples*

Description

Data frame with population triples

Usage

```
example_triples
```

Format

A data frame with population triples, generated by `summarize_triples`

extract_afs *Compute and store blocked allele frequency data*

Description

Prepare data for various *ADMIXTOOLS 2* functions. Reads data from packedancestrymap or PLINK files, and computes allele frequencies for selected populations and stores it as `.rds` files in outdir.

Usage

```
extract_afs(
  pref,
  outdir,
  inds = NULL,
  pops = NULL,
  cols_per_chunk = 10,
  numparts = 100,
  maxmiss = 0,
  minmaf = 0,
  maxmaf = 0.5,
  minac2 = FALSE,
  outpop = NULL,
  auto_only = TRUE,
  transitions = TRUE,
  transversions = TRUE,
  keepsnps = NULL,
  format = NULL,
  poly_only = FALSE,
  adjust_pseudohaploid = TRUE,
  verbose = TRUE
)
```

Arguments

pref	Prefix of *PLINK/EIGENSTRAT/PACKEDANCESTRYMAP* files. *EIGENSTRAT/PACKEDANCESTRYMAP* have to end in `.geno`, `.snp`, `.ind`, *PLINK* has to end in `.bed`, `.bim`, `.fam`
outdir	Directory where data will be stored.
inds	Individuals for which data should be extracted
pops	Populations for which data should be extracted. If both `pops` and `inds` are provided, they should have the same length and will be matched by position. If only `pops` is provided, all individuals from the `.ind` or `.fam` file in those populations will be extracted. If only `inds` is provided, each indivdual will be assigned to its own population of the same name. If neither `pops` nor `inds` is provided, all individuals and populations in the `.ind` or `.fam` file will be extracted.
cols_per_chunk	Number of populations per chunk. Lowering this number will lower the memory requirements when running afs_to_f2, but more chunk pairs will have to be computed.
numparts	Number of parts in which genotype data will be read for computing allele frequencies
maxmiss	Discard SNPs which are missing in a fraction of populations higher than `maxmiss`
minmaf	Discard SNPs with minor allele frequency less than `minmaf`
maxmaf	Discard SNPs with minor allele frequency greater than than `maxmaf`
minac2	Discard SNPs with allele count lower than 2 in any population (default `FALSE`). This option should be set to `TRUE` when computing f3-statistics where one population consists mostly of pseudohaploid samples. Otherwise heterozygosity estimates and thus f3-estimates can be biased. `minac2 == 2` will discard SNPs with allele count lower than 2 in any non-singleton population (this option is experimental and is based on the hypothesis that using SNPs with allele count lower than 2 only leads to biases in non-singleton populations). While the `minac2` option discards SNPs with allele count lower than 2 in any population, the qp3pop function will only discard SNPs with allele count lower than 2 in the first (target) population (when the first argument is the prefix of a genotype file).
outpop	Keep only SNPs which are heterozygous in this population
auto_only	Keep only SNPs on chromosomes 1 to 22
transitions	Set this to `FALSE` to exclude transition SNPs
transversions	Set this to `FALSE` to exclude transversion SNPs
keepsnps	SNP IDs of SNPs to keep. Overrides other SNP filtering options
format	Supply this if the prefix can refer to genotype data in different formats and you want to choose which one to read. Should be `plink` to read `.bed`, `.bim`, `.fam` files, or `eigenstrat`, or `packedancestrymap` to read `.geno`, `.snp`, `.ind` files.

poly_only — Specify whether SNPs with identical allele frequencies in every population should be discarded (poly_only = TRUE), or whether they should be used (poly_only = FALSE). By default (poly_only = c("f2")), these SNPs will be used to compute FST and allele frequency products, but not to compute f2 (this is the default option in the original ADMIXTOOLS).

adjust_pseudohaploid — Genotypes of pseudohaploid samples are usually coded as 0 or 2, even though only one allele is observed. adjust_pseudohaploid ensures that the observed allele count increases only by 1 for each pseudohaploid sample. If TRUE (default), samples that don't have any genotypes coded as 1 among the first 1000 SNPs are automatically identified as pseudohaploid. This leads to slightly more accurate estimates of f-statistics. Setting this parameter to FALSE treats all samples as diploid and is equivalent to the *ADMIXTOOLS* inbreed: NO option. Setting adjust_pseudohaploid to an integer n will check the first n SNPs instead of the first 1000 SNPs.

verbose — Print progress updates

Value

SNP metadata (invisibly)

Examples

```
## Not run:
pref = 'my/genofiles/prefix'
outdir = 'dir/for/afdata/'
extract_afs(pref, outdir)

## End(Not run)
```

---

extract_afs_simple *Compute and store blocked allele frequency data*

Description

Prepare data for various *ADMIXTOOLS 2* functions. Reads data from packedancestrymap or PLINK files, and computes allele frequencies for selected populations and stores it as .rds files in outdir.

Usage

```
extract_afs_simple(
  pref,
  outdir,
  inds = NULL,
  pops = NULL,
  blgsize = 0.05,
  cols_per_chunk = 10,
```

```
  maxmiss = 0,
  minmaf = 0,
  maxmaf = 0.5,
  minac2 = FALSE,
  outpop = NULL,
  transitions = TRUE,
  transversions = TRUE,
  keepsnps = NULL,
  format = NULL,
  poly_only = FALSE,
  adjust_pseudohaploid = TRUE,
  verbose = TRUE
)
```

Arguments

`pref`	Prefix of *PLINK/EIGENSTRAT/PACKEDANCESTRYMAP* files. *EIGENSTRAT/PACKEDANCESTRYMAP* have to end in `.geno`, `.snp`, `.ind`, *PLINK* has to end in `.bed`, `.bim`, `.fam`
`outdir`	Directory where data will be stored.
`inds`	Individuals for which data should be extracted
`pops`	Populations for which data should be extracted. If both `pops` and `inds` are provided, they should have the same length and will be matched by position. If only `pops` is provided, all individuals from the `.ind` or `.fam` file in those populations will be extracted. If only `inds` is provided, each indivdual will be assigned to its own population of the same name. If neither `pops` nor `inds` is provided, all individuals and populations in the `.ind` or `.fam` file will be extracted.
`blgsize`	SNP block size in Morgan. Default is 0.05 (5 cM). If `blgsize` is 100 or greater, if will be interpreted as base pair distance rather than centimorgan distance.
`cols_per_chunk`	Number of populations per chunk. Lowering this number will lower the memory requirements when running `afs_to_f2`, but more chunk pairs will have to be computed.
`maxmiss`	Discard SNPs which are missing in a fraction of populations higher than `maxmiss`
`minmaf`	Discard SNPs with minor allele frequency less than `minmaf`
`maxmaf`	Discard SNPs with minor allele frequency greater than than `maxmaf`
`minac2`	Discard SNPs with allele count lower than 2 in any population (default `FALSE`). This option should be set to `TRUE` when computing f3-statistics where one population consists mostly of pseudohaploid samples. Otherwise heterozygosity estimates and thus f3-estimates can be biased. `minac2 == 2` will discard SNPs with allele count lower than 2 in any non-singleton population (this option is experimental and is based on the hypothesis that using SNPs with allele count lower than 2 only leads to biases in

non-singleton populations). While the `minac2` option discards SNPs with allele count lower than 2 in any population, the `qp3pop` function will only discard SNPs with allele count lower than 2 in the first (target) population (when the first argument is the prefix of a genotype file).

`outpop` Keep only SNPs which are heterozygous in this population

`transitions` Set this to `FALSE` to exclude transition SNPs

`transversions` Set this to `FALSE` to exclude transversion SNPs

`keepsnps` SNP IDs of SNPs to keep. Overrides other SNP filtering options

`format` Supply this if the prefix can refer to genotype data in different formats and you want to choose which one to read. Should be `plink` to read `.bed`, `.bim`, `.fam` files, or `eigenstrat`, or `packedancestrymap` to read `.geno`, `.snp`, `.ind` files.

`poly_only` Specify whether SNPs with identical allele frequencies in every population should be discarded (`poly_only = TRUE`), or whether they should be used (`poly_only = FALSE`). By default (`poly_only = c("f2")`), these SNPs will be used to compute FST and allele frequency products, but not to compute f2 (this is the default option in the original ADMIXTOOLS).

`adjust_pseudohaploid`
Genotypes of pseudohaploid samples are usually coded as `0` or `2`, even though only one allele is observed. `adjust_pseudohaploid` ensures that the observed allele count increases only by `1` for each pseudohaploid sample. If `TRUE` (default), samples that don't have any genotypes coded as `1` among the first 1000 SNPs are automatically identified as pseudohaploid. This leads to slightly more accurate estimates of f-statistics. Setting this parameter to `FALSE` treats all samples as diploid and is equivalent to the *ADMIXTOOLS* `inbreed: NO` option. Setting `adjust_pseudohaploid` to an integer `n` will check the first `n` SNPs instead of the first 1000 SNPs.

`verbose` Print progress updates

Value

SNP metadata (invisibly)

Examples

```
## Not run:
pref = 'my/genofiles/prefix'
outdir = 'dir/for/afdata/'
extract_afs(pref, outdir)

## End(Not run)
```

`extract_counts` *Extract and store data needed to compute blocked f2*

Description

Prepare data for various *ADMIXTOOLS 2* functions. This function reads data from genotype files, and extracts data required to compute blocked f-statistics for any sets of samples. The data consists of `.rds` files with total and alternative allele counts for each individual, and products of total and alternative allele counts for each pair. The function calls `packedancestrymap_to_afs` or `plink_to_afs` and `afs_to_f2_blocks`.

Usage

```
extract_counts(
  pref,
  outdir,
  inds = NULL,
  blgsize = 0.05,
  maxmiss = 0,
  minmaf = 0,
  maxmaf = 0.5,
  transitions = TRUE,
  transversions = TRUE,
  auto_only = TRUE,
  keepsnps = NULL,
  maxmem = 8000,
  overwrite = FALSE,
  format = NULL,
  cols_per_chunk = NULL,
  verbose = TRUE
)
```

Arguments

`pref`	Prefix of *PLINK/EIGENSTRAT/PACKEDANCESTRYMAP* files. *EIGENSTRAT/PACKEDANCESTRYMAP* have to end in `.geno`, `.snp`, `.ind`, *PLINK* has to end in `.bed`, `.bim`, `.fam`
`outdir`	Directory where data will be stored.
`inds`	Individuals for which data should be read. Defaults to all individuals
`blgsize`	SNP block size in Morgan. Default is 0.05 (5 cM). If `blgsize` is 100 or greater, if will be interpreted as base pair distance rather than centimorgan distance.
`maxmiss`	Discard SNPs which are missing in a fraction of individuals greater than `maxmiss`
`minmaf`	Discard SNPs with minor allele frequency less than `minmaf`

`maxmaf`	Discard SNPs with minor allele frequency greater than than `maxmaf`
`transitions`	Set this to `FALSE` to exclude transition SNPs
`transversions`	Set this to `FALSE` to exclude transversion SNPs
`auto_only`	Keep only SNPs on chromosomes 1 to 22
`keepsnps`	SNP IDs of SNPs to keep. Overrides other SNP filtering options
`maxmem`	Maximum amount of memory to be used. If the required amount of memory exceeds `maxmem`, allele frequency data will be split into blocks, and the computation will be performed separately on each block pair. This doesn't put a precise cap on the amount of memory used (it used to at some point). Set this parameter to lower values if you run out of memory while running this function. Set it to higher values if this function is too slow and you have lots of memory.
`overwrite`	Overwrite existing files in `outdir`
`format`	Supply this if the prefix can refer to genotype data in different formats and you want to choose which one to read. Should be `plink` to read `.bed`, `.bim`, `.fam` files, or `eigenstrat`, or `packedancestrymap` to read `.geno`, `.snp`, `.ind` files.
`cols_per_chunk`	Number of genotype chunks to store on disk. Setting this to a positive integer makes the function slower, but requires less memory. The default value for `cols_per_chunk` in `extract_afs` is 10. Lower numbers will lower the memory requirement but increase the time it takes.
`verbose`	Print progress updates

`extract_f2` *Compute and store blocked f2 statistics*

Description

This function prepares data for various other *ADMIXTOOLS 2* functions. It reads data from genotype files, computes allele frequencies and blocked f2-statistics for selected populations, and writes the results to `outdir`.

Usage

```
extract_f2(
  pref,
  outdir,
  inds = NULL,
  pops = NULL,
  blgsize = 0.05,
  maxmem = 8000,
  maxmiss = 0,
  minmaf = 0,
```

```
  maxmaf = 0.5,
  minac2 = FALSE,
  pops2 = NULL,
  outpop = NULL,
  outpop_scale = TRUE,
  transitions = TRUE,
  transversions = TRUE,
  auto_only = TRUE,
  keepsnps = NULL,
  overwrite = FALSE,
  format = NULL,
  adjust_pseudohaploid = TRUE,
  cols_per_chunk = NULL,
  fst = TRUE,
  afprod = TRUE,
  poly_only = c("f2"),
  apply_corr = TRUE,
  qpfstats = FALSE,
  n_cores = 1,
  verbose = TRUE,
  ...
)
```

Arguments

`pref`	Prefix of *PLINK/EIGENSTRAT/PACKEDANCESTRYMAP* files. *EIGENSTRAT/PACKEDANCESTRYMAP* have to end in `.geno`, `.snp`, `.ind`, *PLINK* has to end in `.bed`, `.bim`, `.fam`
`outdir`	Directory where data will be stored.
`inds`	Individuals for which data should be extracted
`pops`	Populations for which data should be extracted. If both `pops` and `inds` are provided, they should have the same length and will be matched by position. If only `pops` is provided, all individuals from the `.ind` or `.fam` file in those populations will be extracted. If only `inds` is provided, each indivdual will be assigned to its own population of the same name. If neither `pops` nor `inds` is provided, all individuals and populations in the `.ind` or `.fam` file will be extracted.
`blgsize`	SNP block size in Morgan. Default is 0.05 (5 cM). If `blgsize` is 100 or greater, if will be interpreted as base pair distance rather than centimorgan distance.
`maxmem`	Maximum amount of memory to be used. If the required amount of memory exceeds `maxmem`, allele frequency data will be split into blocks, and the computation will be performed separately on each block pair. This doesn't put a precise cap on the amount of memory used (it used to at some point). Set this parameter to lower values if you run out of memory while running this function. Set it to higher values if this function is too slow and you have lots of memory.

`maxmiss` Discard SNPs which are missing in a fraction of populations higher than `maxmiss`

`minmaf` Discard SNPs with minor allele frequency less than `minmaf`

`maxmaf` Discard SNPs with minor allele frequency greater than than `maxmaf`

`minac2` Discard SNPs with allele count lower than 2 in any population (default `FALSE`). This option should be set to `TRUE` when computing f3-statistics where one population consists mostly of pseudohaploid samples. Otherwise heterozygosity estimates and thus f3-estimates can be biased. `minac2 == 2` will discard SNPs with allele count lower than 2 in any non-singleton population (this option is experimental and is based on the hypothesis that using SNPs with allele count lower than 2 only leads to biases in non-singleton populations). While the `minac2` option discards SNPs with allele count lower than 2 in any population, the `qp3pop` function will only discard SNPs with allele count lower than 2 in the first (target) population (when the first argument is the prefix of a genotype file).

`pops2` If specified, only a pairs between `pops` and `pops2` will be computed

`outpop` Keep only SNPs which are heterozygous in this population

`outpop_scale` Scale f2-statistics by the inverse `outpop` heteroygosity (`1/(p*(1-p))`). Providing `outpop` and setting `outpop_scale` to `TRUE` will give the same results as the original *qpGraph* when the `outpop` parameter has been set, but it has the disadvantage of treating one population different from the others. This may limit the use of these f2-statistics for other models.

`transitions` Set this to `FALSE` to exclude transition SNPs

`transversions` Set this to `FALSE` to exclude transversion SNPs

`auto_only` Keep only SNPs on chromosomes 1 to 22

`keepsnps` SNP IDs of SNPs to keep. Overrides other SNP filtering options

`overwrite` Overwrite existing files in `outdir`

`format` Supply this if the prefix can refer to genotype data in different formats and you want to choose which one to read. Should be `plink` to read `.bed`, `.bim`, `.fam` files, or `eigenstrat`, or `packedancestrymap` to read `.geno`, `.snp`, `.ind` files.

`adjust_pseudohaploid`
Genotypes of pseudohaploid samples are usually coded as `0` or `2`, even though only one allele is observed. `adjust_pseudohaploid` ensures that the observed allele count increases only by `1` for each pseudohaploid sample. If `TRUE` (default), samples that don't have any genotypes coded as `1` among the first 1000 SNPs are automatically identified as pseudohaploid. This leads to slightly more accurate estimates of f-statistics. Setting this parameter to `FALSE` treats all samples as diploid and is equivalent to the *ADMIXTOOLS* `inbreed: NO` option. Setting `adjust_pseudohaploid` to an integer `n` will check the first `n` SNPs instead of the first 1000 SNPs.

`cols_per_chunk`
Number of allele frequency chunks to store on disk. Setting this to a positive integer makes the function slower, but requires less memory. The default value for `cols_per_chunk` in `extract_afs` is 10. Lower numbers will lower the memory requirement but increase the time it takes.

`fst`	Write files with pairwise FST for every population pair. Setting this to FALSE can make `extract_f2` faster and will require less memory.
`afprod`	Write files with allele frequency products for every population pair. Setting this to FALSE can make `extract_f2` faster and will require less memory.
`poly_only`	Specify whether SNPs with identical allele frequencies in every population should be discarded (`poly_only = TRUE`), or whether they should be used (`poly_only = FALSE`). By default (`poly_only = c("f2")`), these SNPs will be used to compute FST and allele frequency products, but not to compute f2 (this is the default option in the original ADMIXTOOLS).
`apply_corr`	Apply small-sample-size correction when computing f2-statistics (default `TRUE`)
`qpfstats`	Compute smoothed f2-statistics (default `FALSE`). In the presence of large amounts of missing data, this option can be used to retain information from all SNPs while introducing less bias than setting `maxmiss` to values greater than 0. When setting `qpfstats = TRUE`, most other options to `extract_f2` will be ignored. See `qpfstats` for more information. Arguments to `qpfstats` can be passed via `...`
`n_cores`	Parallelize computation across `n_cores` cores via the `doParallel` package.
`verbose`	Print progress updates
`...`	Pass arguments to `qpfstats`

Value

SNP metadata (invisibly)

See Also

`f2_from_precomp` for reading the stored f2-statistics back into R, `f2_from_geno` to skip writting f2-statistics to disk and return them directly

Examples

```
## Not run:
pref = 'my/genofiles/prefix'
f2dir = 'my/f2dir/'
extract_f2(pref, f2dir, pops = c('popA', 'popB', 'popC'))

## End(Not run)
```

`extract_f2_large` *Compute and store blocked f2 statistics*

Description

`extract_f2_large` does the same as `extract_f2`, but it requires less memory and is slower. `outdir` has to be set in `extract_f2_large`.

Usage

```
extract_f2_large(
  pref,
  outdir,
  inds = NULL,
  pops = NULL,
  blgsize = 0.05,
  cols_per_chunk = 10,
  maxmiss = 0,
  minmaf = 0,
  maxmaf = 0.5,
  minac2 = FALSE,
  outpop = NULL,
  outpop_scale = TRUE,
  transitions = TRUE,
  transversions = TRUE,
  keepsnps = NULL,
  snpblocks = NULL,
  overwrite = FALSE,
  format = NULL,
  adjust_pseudohaploid = TRUE,
  afprod = TRUE,
  fst = TRUE,
  poly_only = c("f2"),
  apply_corr = TRUE,
  verbose = TRUE
)
```

Arguments

`pref`	Prefix of *PLINK/EIGENSTRAT/PACKEDANCESTRYMAP* files. *EIGENSTRAT/PACKEDANCESTRYMAP* have to end in `.geno`, `.snp`, `.ind`, *PLINK* has to end in `.bed`, `.bim`, `.fam`
`outdir`	Directory where data will be stored.
`inds`	Individuals for which data should be extracted
`pops`	Populations for which data should be extracted. If both `pops` and `inds` are provided, they should have the same length and will be matched by

position. If only `pops` is provided, all individuals from the `.ind` or `.fam` file in those populations will be extracted. If only `inds` is provided, each indivdual will be assigned to its own population of the same name. If neither `pops` nor `inds` is provided, all individuals and populations in the `.ind` or `.fam` file will be extracted.

`blgsize` SNP block size in Morgan. Default is 0.05 (5 cM). If `blgsize` is 100 or greater, if will be interpreted as base pair distance rather than centimorgan distance.

`cols_per_chunk`
Number of populations per chunk. Lowering this number will lower the memory requirements when running

`maxmiss` Discard SNPs which are missing in a fraction of populations higher than `maxmiss`

`minmaf` Discard SNPs with minor allele frequency less than `minmaf`

`maxmaf` Discard SNPs with minor allele frequency greater than than `maxmaf`

`minac2` Discard SNPs with allele count lower than 2 in any population (default `FALSE`). This option should be set to `TRUE` when computing f3-statistics where one population consists mostly of pseudohaploid samples. Otherwise heterozygosity estimates and thus f3-estimates can be biased. `minac2 == 2` will discard SNPs with allele count lower than 2 in any non-singleton population (this option is experimental and is based on the hypothesis that using SNPs with allele count lower than 2 only leads to biases in non-singleton populations). While the `minac2` option discards SNPs with allele count lower than 2 in any population, the qp3pop function will only discard SNPs with allele count lower than 2 in the first (target) population (when the first argument is the prefix of a genotype file).

`outpop` Keep only SNPs which are heterozygous in this population

`outpop_scale` Scale f2-statistics by the inverse `outpop` heteroygosity (`1/(p*(1-p))`). Providing `outpop` and setting `outpop_scale` to `TRUE` will give the same results as the original *qpGraph* when the `outpop` parameter has been set, but it has the disadvantage of treating one population different from the others. This may limit the use of these f2-statistics for other models.

`transitions` Set this to `FALSE` to exclude transition SNPs

`transversions` Set this to `FALSE` to exclude transversion SNPs

`keepsnps` SNP IDs of SNPs to keep. Overrides other SNP filtering options

`snpblocks` Optional named integer vector assigning each SNP to a block; if `NULL`, blocks are computed from `blgsize`.

`overwrite` Overwrite existing files in `outdir`

`format` Supply this if the prefix can refer to genotype data in different formats and you want to choose which one to read. Should be `plink` to read `.bed`, `.bim`, `.fam` files, or `eigenstrat`, or `packedancestrymap` to read `.geno`, `.snp`, `.ind` files.

`adjust_pseudohaploid`
Genotypes of pseudohaploid samples are usually coded as `0` or `2`, even though only one allele is observed. `adjust_pseudohaploid` ensures that

the observed allele count increases only by `1` for each pseudohaploid sample. If `TRUE` (default), samples that don't have any genotypes coded as `1` among the first 1000 SNPs are automatically identified as pseudohaploid. This leads to slightly more accurate estimates of f-statistics. Setting this parameter to `FALSE` treats all samples as diploid and is equivalent to the *ADMIXTOOLS* `inbreed: NO` option. Setting `adjust_pseudohaploid` to an integer `n` will check the first `n` SNPs instead of the first 1000 SNPs.

`afprod` Write files with allele frequency products for every population pair. Setting this to FALSE can make `extract_f2` faster and will require less memory.

`fst` Write files with pairwise FST for every population pair. Setting this to FALSE can make `extract_f2` faster and will require less memory.

`poly_only` Specify whether SNPs with identical allele frequencies in every population should be discarded (`poly_only = TRUE`), or whether they should be used (`poly_only = FALSE`). By default (`poly_only = c("f2")`), these SNPs will be used to compute FST and allele frequency products, but not to compute f2 (this is the default option in the original ADMIXTOOLS).

`apply_corr` Apply small-sample-size correction when computing f2-statistics (default `TRUE`)

`verbose` Print progress updates

Details

`extract_f2_large` requires less memory because it writes allele frequency data to disk, and doesn't store the allele frequency matrix for all populations and SNPs in memory. If you still run out of memory, reduce `cols_per_chunk`. This function is a wrapper around `extract_afs` and `afs_to_f2`, and is slower than `extract_f2`. It may be faster to call `extract_afs` and `afs_to_f2` directly, parallelizing over the different calls to `afs_to_f2`.

Value

SNP metadata (invisibly)

See Also

`extract_f2`

Examples

```
## Not run:
pref = 'my/genofiles/prefix'
f2dir = 'my/f2dir/'
extract_f2_large(pref, f2dir, pops = c('popA', 'popB', 'popC'))

## End(Not run)
```

extract_f2_subset *Copy f2-statistics*

Description

Copy a subset of f2-statistics to a new directory

Usage

```
extract_f2_subset(from, to, pops, verbose = TRUE)
```

Arguments

from	Directory with f2-statistics
to	Target directory
pops	Populations to copy
verbose	Print progress updates

Examples

```
## Not run:
pref = 'my/genofiles/prefix'
outdir = 'dir/for/afdata/'
extract_f2_subset(pref, outdir)

## End(Not run)
```

extract_samples *Extract samples from PLINK files*

Description

This function reads *PLINK* files, extracts a subset of samples, and writes new *PLINK* files using write_plink. It's probably slower than running the equivalent command in *PLINK* directly, but it can be useful to do this from within R. When inds or pops is provided, only a subset of samples will be extracted.

Usage

```
extract_samples(
  inpref,
  outpref,
  inds = NULL,
  pops = NULL,
  overwrite = FALSE,
  verbose = TRUE
)
```

Arguments

`inpref`	Prefix of the *PLINK* input files
`outpref`	Prefix of the *PLINK* output files
`inds`	Individuals which should be extracted
`pops`	Populations which should be extracted. Can not be provided together with `inds`
`overwrite`	Set this to `TRUE` if `inpref == outpref` and you really want to overwrite the input files.
`verbose`	Print progress updates

`f2` *Estimate f2 statistics*

Description

Computes f2 statistics from f2 blocks of the form $f2(A, B)$

Usage

```
f2(
  data,
  pop1 = NULL,
  pop2 = NULL,
  boot = FALSE,
  sure = FALSE,
  unique_only = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

`data`	Input data in one of three forms: 1. A 3d array of blocked f2 statistics, output of `f2_from_precomp` or `f2_from_geno` (fastest option) 2. A directory which contains pre-computed f2-statistics 3. The prefix of genotype files (slowest option)
`pop1`	One of the following four: 1. `NULL`: all possible population combinations will be returned 2. A vector of population labels. All combinations with the other `pop` arguments will be returned 3. A matrix with population combinations to be tested, with one population per column and one combination per row. Other `pop` arguments will be ignored.

4. the location of a file (`poplistname` or `popfilename`) which specifies the populations or population combinations to be tested. Other `pop` arguments will be ignored.

`pop2`	A vector of population labels
`boot`	If `FALSE` (the default), block-jackknife resampling will be used to compute standard errors. Otherwise, block-bootstrap resampling will be used to compute standard errors. If `boot` is an integer, that number will specify the number of bootstrap resamplings. If `boot = TRUE`, the number of bootstrap resamplings will be equal to the number of SNP blocks.
`sure`	The number of population combinations can get very large. This is a safety option that stops you from accidently computing all combinations if that number is large.
`unique_only`	If `TRUE` (the default), redundant combinations will be excluded
`verbose`	Print progress updates
`...`	Additional arguments passed to `f2_from_geno` when `data` is a genotype prefix

Value

`f2` returns a data frame with f2 statistics

References

Patterson, N. et al. (2012) *Ancient admixture in human history* Genetics

Peter, B. (2016) *Admixture, Population Structure, and F-Statistics* Genetics

Examples

```
pop1 = 'Denisova.DG'
pop2 = c('Altai_Neanderthal.DG', 'Vindija.DG')
f2(example_f2_blocks, pop1, pop2)
## Not run:
f2(f2_dir, pop1, pop2)

## End(Not run)
```

`f2_from_geno` *Compute blocked f2 statistics*

Description

This function prepares data for various other *ADMIXTOOLS 2* functions. It reads data from genotype files, computes allele frequencies and blocked f2-statistics for selected populations, and returns them as a 3d array.

Usage

```
f2_from_geno(
  pref,
  inds = NULL,
  pops = NULL,
  blgsize = 0.05,
  maxmem = 8000,
  maxmiss = 0,
  minmaf = 0,
  maxmaf = 0.5,
  pops2 = NULL,
  outpop = NULL,
  outpop_scale = TRUE,
  transitions = TRUE,
  transversions = TRUE,
  auto_only = TRUE,
  keepsnps = NULL,
  afprod = FALSE,
  fst = FALSE,
  poly_only = c("f2"),
  format = NULL,
  adjust_pseudohaploid = TRUE,
  remove_na = TRUE,
  apply_corr = TRUE,
  qpfstats = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`pref`	Prefix of *PLINK/EIGENSTRAT/PACKEDANCESTRYMAP* files. *EIGENSTRAT/PACKEDANCESTRYMAP* have to end in `.geno`, `.snp`, `.ind`, *PLINK* has to end in `.bed`, `.bim`, `.fam`
`inds`	Individuals for which data should be extracted
`pops`	Populations for which data should be extracted. If both `pops` and `inds` are provided, they should have the same length and will be matched by position. If only `pops` is provided, all individuals from the `.ind` or `.fam` file in those populations will be extracted. If only `inds` is provided, each indivdual will be assigned to its own population of the same name. If neither `pops` nor `inds` is provided, all individuals and populations in the `.ind` or `.fam` file will be extracted.
`blgsize`	SNP block size in Morgan. Default is 0.05 (5 cM). If `blgsize` is 100 or greater, if will be interpreted as base pair distance rather than centimorgan distance.
`maxmem`	Maximum amount of memory to be used. If the required amount of memory exceeds `maxmem`, allele frequency data will be split into blocks,

and the computation will be performed separately on each block pair. This doesn't put a precise cap on the amount of memory used (it used to at some point). Set this parameter to lower values if you run out of memory while running this function. Set it to higher values if this function is too slow and you have lots of memory.

`maxmiss` Discard SNPs which are missing in a fraction of populations higher than `maxmiss`

`minmaf` Discard SNPs with minor allele frequency less than `minmaf`

`maxmaf` Discard SNPs with minor allele frequency greater than than `maxmaf`

`pops2` If specified, only a pairs between `pops` and `pops2` will be computed

`outpop` Keep only SNPs which are heterozygous in this population

`outpop_scale` Scale f2-statistics by the inverse `outpop` heteroygosity (`1/(p*(1-p))`). Providing `outpop` and setting `outpop_scale` to `TRUE` will give the same results as the original *qpGraph* when the `outpop` parameter has been set, but it has the disadvantage of treating one population different from the others. This may limit the use of these f2-statistics for other models.

`transitions` Set this to `FALSE` to exclude transition SNPs

`transversions` Set this to `FALSE` to exclude transversion SNPs

`auto_only` Keep only SNPs on chromosomes 1 to 22

`keepsnps` SNP IDs of SNPs to keep. Overrides other SNP filtering options

`afprod` Return negative average allele frequency products instead of f2-statistics. Setting `afprod = TRUE` will result in more precise f4-statistics when the original data had large amounts of missingness, and should be used in that case for `qpdstat` and `qpadm`. It can also be used for outgroup f3-statistics with a fixed outgroup (for example for `qpgraph`); values will be shifted by a constant amount compared to regular f3-statistics. This shift affects the fit of a graph only by small amounts, possibly less than bias in regular f3-statistics introduced by large amounts of missing data.

`fst` Write files with pairwise FST for every population pair. Setting this to FALSE can make `extract_f2` faster and will require less memory.

`poly_only` Specify whether SNPs with identical allele frequencies in every population should be discarded (`poly_only = TRUE`), or whether they should be used (`poly_only = FALSE`). By default (`poly_only = c("f2")`), these SNPs will be used to compute FST and allele frequency products, but not to compute f2 (this is the default option in the original ADMIXTOOLS).

`format` Supply this if the prefix can refer to genotype data in different formats and you want to choose which one to read. Should be `plink` to read `.bed`, `.bim`, `.fam` files, or `eigenstrat`, or `packedancestrymap` to read `.geno`, `.snp`, `.ind` files.

`adjust_pseudohaploid`
Genotypes of pseudohaploid samples are usually coded as `0` or `2`, even though only one allele is observed. `adjust_pseudohaploid` ensures that the observed allele count increases only by `1` for each pseudohaploid sample. If `TRUE` (default), samples that don't have any genotypes coded as `1`

	among the first 1000 SNPs are automatically identified as pseudohaploid. This leads to slightly more accurate estimates of f-statistics. Setting this parameter to `FALSE` treats all samples as diploid and is equivalent to the *ADMIXTOOLS* `inbreed: NO` option. Setting `adjust_pseudohaploid` to an integer `n` will check the first `n` SNPs instead of the first 1000 SNPs.
`remove_na`	Remove blocks with missing values (default `TRUE`).
`apply_corr`	Apply small-sample-size correction when computing f2-statistics (default `TRUE`)
`qpfstats`	Compute smoothed f2-statistics (default `FALSE`). In the presence of large amounts of missing data, this option can be used to retain information from all SNPs while introducing less bias than setting `maxmiss` to values greater than 0. When setting `qpfstats = TRUE`, most other options to `extract_f2` will be ignored. See `qpfstats` for more information. Arguments to `qpfstats` can be passed via `...`
`verbose`	Print progress updates
`...`	Pass arguments to `qpfstats`

Value

A 3d array of f2-statistics (or scaled allele frequency products if `afprod = TRUE`)

See Also

`f2_from_precomp` for reading previously stored f2-statistics into R, `extract_f2` for storing f2-statistics on disk

---

f2_from_msprime *Simulate an admixture graph in msprime*

Description

This function generates an msprime simulation script, executes it in python, and turns the resulting genotype data into f2-statistics

Usage

```
f2_from_msprime(..., blgsize = 0.05, cleanup = TRUE, verbose = TRUE)
```

Arguments

`...`	Arguments passed to `msprime_sim`.
`blgsize`	SNP block size in Morgan for computing f2-statistics (default 0.05).
`cleanup`	Delete temporary simulation files after reading (default `TRUE`).
`verbose`	Print progress updates.

See Also

`msprime_sim`

`f2_from_precomp` *Read blocked f2 statistics from disk*

Description

Read blocked f2 statistics from disk

Usage

```
f2_from_precomp(
  dir,
  inds = NULL,
  pops = NULL,
  pops2 = NULL,
  afprod = FALSE,
  fst = FALSE,
  return_array = TRUE,
  apply_corr = TRUE,
  remove_na = TRUE,
  verbose = TRUE
)
```

Arguments

`dir`	Directory with precomputed f2 statistics, or precomputed individual pair data
`inds`	Individuals for which data should be read. Defaults to all individuals, which may require a lot of memory.
`pops`	Populations for which data should be read. Defaults to all populations, which may require a lot of memory.
`pops2`	Optional second vector of populations. Useful if a f4 statistics of a few against many populations should be computed. `pops2` should not be specified in other cases, as most functions depend on f2-statistics for all population pairs in `pops`.
`afprod`	Return negative average allele frequency products instead of f2 estimates. This will result in more precise f4-statistics when the original data had large amounts of missingness, and should be used in that case for `qpdstat` and `qpadm`. It can also be used for outgroup f3-statistics with a fixed outgroup (for example for `qpgraph`); values will be shifted by a constant amount compared to regular f3-statistics. This shift affects the fit of a graph only by small amounts, possibly less than bias in regular f3-statistics introduced by large amounts of missing data. This option is currently ineffective when reading data extracted with `extract_counts`.
`fst`	Read FST statistics instead of f2 (default `FALSE`).
`return_array`	Return a 3d array (default). If false, a data frame will be returned.

`apply_corr`	Subtract the f2 correction factor. Setting this to `FALSE` can occasionally be useful
`remove_na`	Remove blocks with missing values
`verbose`	Print progress updates

Value

A 3d array of f2 statistics

Examples

```
## Not run:
dir = 'my/f2/dir/'
f2_blocks = f2_from_precomp(dir, pops = c('pop1', 'pop2', 'pop3'))

## End(Not run)
```

`f2dat_f4dat` *Turn f2 data to f4 data*

Description

Turn f2 data to f4 data

Usage

```
f2dat_f4dat(f2dat, popcomb = NULL)
```

Arguments

`f2dat`	A data frame of f2-statistics with columns `pop1`, `pop2`, `f2`
`popcomb`	Optional data frame specifying which population quadruples to compute; if `NULL`, all quadruples are used.

Value

A data frame with f4-statistics

`f3blockdat_from_geno` *f3 from genotype data*

Description

Compute per-block f3-statistics directly from genotype data

Usage

```
f3blockdat_from_geno(
  pref,
  popcombs,
  auto_only = TRUE,
  blgsize = 0.05,
  block_lengths = NULL,
  allsnps = FALSE,
  adjust_pseudohaploid = TRUE,
  poly_only = FALSE,
  apply_corr = TRUE,
  outgroupmode = FALSE,
  verbose = TRUE
)
```

Arguments

`pref`	Prefix of genotype files
`popcombs`	A data frame with one population combination per row, and columns `pop1`, `pop2`, `pop3`, `pop4`. If there is an additional integer column named `model` and `allsnps = FALSE`, only SNPs present in every population in any given model will be used to compute f4-statistics for that model.
`auto_only`	Use only chromosomes 1 to 22.
`blgsize`	SNP block size in Morgan. Default is 0.05 (5 cM). If `blgsize` is 100 or greater, if will be interpreted as base pair distance rather than centimorgan distance.
`block_lengths`	An optional vector with block lengths. If `NULL`, block lengths will be computed.
`allsnps`	Use all SNPs with allele frequency estimates in every population of any given population quadruple. If `FALSE` (the default) only SNPs which are present in all populations in `popcombs` (or any given model in it) will be used. Setting `allsnps = TRUE` in the presence of large amounts of missing data might lead to false positive results.
`adjust_pseudohaploid`	Genotypes of pseudohaploid samples are usually coded as `0` or `2`, even though only one allele is observed. `adjust_pseudohaploid` ensures that the observed allele count increases only by `1` for each pseudohaploid sample. If `TRUE` (default), samples that don't have any genotypes coded as

1 among the first 1000 SNPs are automatically identified as pseudohaploid. This leads to slightly more accurate estimates of f-statistics. Setting this parameter to FALSE is equivalent to the ADMIXTOOLS inbreed: NO option. Setting adjust_pseudohaploid to an integer n will check the first n SNPs instead of the first 1000 SNPs.

poly_only — Only use polymorphic SNPs (default FALSE).

apply_corr — With apply_corr = FALSE, no bias correction is performed. With apply_corr = TRUE (the default), a bias correction term based on the heterozygosity in the first population is subtracted from the f3 estimate. With apply_corr = 2, the bias correction term is calculated based on all 3 populations. This option is not generally recommended, and only exists to match how the f3-statistics are estimated in certain scenarios in the original qpGraph program.

outgroupmode — With outgroupmode = FALSE, estimates of f3 will be normalized by estimates of the heterozygosity of the target population. This is the default option if the first argument is the prefix of genotype data. If the first argument is an array of precomputed f2-statistics, then no normalization can be performed, which corresponds to outgroupmode = TRUE.

verbose — Print progress updates

Value

A data frame with per-block f4-statistics for each population quadruple.

---

f4_from_afdat *Compute f4 from allele frequencies*

Description

Compute f4 from allele frequencies

Usage

```
f4_from_afdat(afdat, popcombs)
```

Arguments

afdat — A data frame with allele frequencies and SNP metadata. Can be grouped.

popcombs — A data frame with population combinations. Columns pop1 to pop4

Examples

```
## Not run:
# Compute f4 for all mutatation classes separately
afs = plink_to_afs('/my/geno/prefix', pops = c('p1', 'p2', 'p3', 'p4', 'p5'))
afdat = bind_cols(afs$snpfile, afs$afs %>% as_tibble()) %>%
        mutate(gr = paste0(pmin(A1, A2), pmax(A1, A2))) %>%
```

```
        group_by(gr)
popcombs = tibble(pop1 = c('p1', 'p5'), pop2 = 'p2', pop3 = 'p3', pop4 = 'p4')
out = f4_from_afdat(afdat, popcombs)
out %>% ggplot(aes(gr, est)) + geom_point() +
          geom_errorbar(aes(ymin = est - se, ymax = est + se)) +
          facet_wrap(~paste(pop1, pop2, pop3, pop4), scales = 'free')

## End(Not run)
```

`f4_from_f2` *Get per-block f4-statistics*

Description

This function turns per-block f2-statistics into per-block f4-statistics of the form `f4(pop1, pop2; pop3, pop4)`

Usage

```
f4_from_f2(f2_data, pop1, pop2 = NULL, pop3 = NULL, pop4 = NULL)
```

Arguments

`f2_data`	A 3d array with blocked f2 statistics, output of `f2_from_precomp` or `f2_from_geno` Alternatively, a directory with precomputed data. See `extract_f2` and `extract_counts`.
`pop1`	Either the name(s) of the first population(s), or a four column matrix with the names of all four populations.
`pop2`	Population 2 (same length as `pop1`)
`pop3`	Population 3 (same length as `pop1`)
`pop4`	Population 4 (same length as `pop1`)

Value

A matrix of per-block f4-statistics (`popcomb x block`)

`f4blockdat_from_geno` *f4 from genotype data*

Description

Compute per-block f4-statistics directly from genotype data

Usage

```
f4blockdat_from_geno(
  pref,
  popcombs = NULL,
  left = NULL,
  right = NULL,
  auto_only = TRUE,
  blgsize = 0.05,
  block_lengths = NULL,
  f4mode = TRUE,
  allsnps = FALSE,
  poly_only = FALSE,
  snpwt = NULL,
  keepsnps = NULL,
  cm_file = NULL,
  verbose = TRUE,
  return_matrices = FALSE
)
```

Arguments

`pref`	Prefix of genotype files
`popcombs`	A data frame with one population combination per row, and columns `pop1`, `pop2`, `pop3`, `pop4`. If there is an additional integer column named `model` and `allsnps = FALSE`, only SNPs present in every population in any given model will be used to compute f4-statistics for that model.
`left`	Populations on the left side of f4 (`pop1` and `pop2`). Can be provided together with `right` in place of `popcombs`.
`right`	Populations on the right side of f4 (`pop3` and `pop4`). Can be provided together with `left` in place of `popcombs`.
`auto_only`	Use only chromosomes 1 to 22.
`blgsize`	SNP block size in Morgan. Default is 0.05 (5 cM). If `blgsize` is 100 or greater, if will be interpreted as base pair distance rather than centimorgan distance.
`block_lengths`	An optional vector with block lengths. If `NULL`, block lengths will be computed.
`f4mode`	If `TRUE`: f4 is computed from allele frequencies `a`, `b`, `c`, and `d` as `(a-b)*(c-d)`. if `FALSE`, D-statistics are computed instead, defined as `(a-b)*(c-d) / ((a + b - 2*a*b) * (c + d - 2*c*d))`, which is the same as `(P(BABA) - P(ABBA)) / (P(ABBA) + P(BABA))`.
`allsnps`	Use all SNPs with allele frequency estimates in every population of any given population quadruple. If `FALSE` (the default) only SNPs which are present in all populations in `popcombs` (or any given model in it) will be used. Setting `allsnps = TRUE` in the presence of large amounts of missing data might lead to false positive results.

`poly_only` Only keep SNPs with mean allele frequency not equal to 0 or 1 (default `FALSE`).

`snpwt` A vector of SNP weights

`keepsnps` A vector of SNP IDs to keep

`cm_file` Optional path to a TSV file with per-variant centimorgan positions (columns: a SNP-identifier column and `cm`). Overrides cM values in `.pvar` when supplied.

`verbose` Print progress updates

`return_matrices` If `TRUE`, return a list with `numer`, `cnt`, `denom`, and `block_lengths` matrices directly (the matrices the per- block reducer writes into) instead of the long-format data frame. Used by `qpfstats()` to skip the costly long-format expansion and immediate collapse back to matrices. Default `FALSE` preserves the historical return shape for all other callers.

Value

A data frame with per-block f4-statistics for each population quadruple.

`f4blockdat_to_f4blocks`

Turn f4 block data to 3d array

Description

Turn f4 block data to 3d array

Usage

```
f4blockdat_to_f4blocks(f4blockdat, remove_na = TRUE)
```

Arguments

`f4blockdat` f4 block data frame generated by `f4blockdat_from_geno`

`remove_na` Remove blocks with missing values

find_admixedges *Find admixture edges*

Description

Find admixture edges

Usage

```
find_admixedges(graph)
```

Arguments

graph	An admixture graph

Value

A data frame with columns `from` and `to` with admixture edges

See Also

`find_normedges` `find_newedges`

find_graphs *Find well fitting admixture graphs*

Description

This function generates and evaluates admixture graphs in `numgen` iterations to find well fitting admixturegraphs.

Usage

```
find_graphs(
  data,
  numadmix = 0,
  outpop = NULL,
  stop_gen = 100,
  stop_gen2 = 15,
  stop_score = 0,
  stop_sec = NULL,
  initgraph = NULL,
  numgraphs = 10,
  mutfuns = namedList(spr_leaves, spr_all, swap_leaves, move_admixedge_once,
    flipadmix_random, place_root_random, mutate_n),
  opt_worst_residual = FALSE,
```

```
  plusminus_generations = 5,
  return_searchtree = FALSE,
  admix_constraints = NULL,
  event_constraints = NULL,
  reject_f4z = 0,
  max_admix = numadmix,
  verbose = TRUE,
  ...
)
```

Arguments

`data`	Input data in one of three forms: 1. A 3d array of blocked f2 statistics, output of `f2_from_precomp` or `f2_from_geno` 2. A directory which contains pre-computed f2-statistics 3. The prefix of genotype files
`numadmix`	Number of admixture events within each graph. (Only relevant if `initgraph = NULL`)
`outpop`	Name of the outgroup population
`stop_gen`	Total number of generations after which to stop
`stop_gen2`	Number of generations without improvement after which to stop
`stop_score`	Stop once this score has been reached
`stop_sec`	Number of seconds after which to stop
`initgraph`	Graph to start with. If it is specified, `numadmix` and `outpop` will be inferred from this graph.
`numgraphs`	Number of graphs in each generation
`mutfuns`	Functions used to modify graphs. Defaults to the following: • `spr_leaves`: Subtree prune and regraft leaves. Cuts a leaf node and attaches it to a random other edge in the graph. • `spr_all`: Subtree prune and regraft. Cuts any edge and attaches the new orphan node to a random other edge in the graph, keeping the number of admixture nodes constant. • `swap_leaves`: Swaps two leaf nodes. • `move_admixedge_once`: Moves an admixture edge to a nearby location. • `flipadmix_random`: Flips the direction of an admixture edge (if possible). • `mutate_n`: Apply `n` of the mutation functions in this list to a graph (defaults to 2).
`opt_worst_residual`	Optimize for lowest worst residual instead of best score. `FALSE` by default, because the likelihood score is generally a better indicator of the quality of the model fit, and because optimizing for the lowest worst residual is slower (because f4-statistics need to be computed).

`plusminus_generations`
: If the best score does not improve after `plusminus_generations` generations, another approach to improving the score will be attempted: A number of graphs with on additional admixture edge will be generated and evaluated. The resulting graph with the best score will be picked, and new graphs will be created by removing any one admixture edge (bringing the number back to what it was originally). The graph with the lowest score will then be selected. This often makes it possible to break out of local optima, but is slower than regular graph modifications. If the current number of admixture events is lower than `max_numadmix`, the last step (removing an admixture edge) will be skipped.

`return_searchtree`
: Return the search tree in addition to the models. Output will be a list with three items: models, search tree, search tree as data frame

`admix_constraints`
: A data frame with constraints on the number of admixture events for each population. See `satisfies_numadmix` As soon as one graph happens to satisfy these constraints, all subsequently generated graphs will be required to also satisfy them.

`event_constraints`
: A data frame with constraints on the order of events in an admixture graph. See `satisfies_eventorder` As soon as one graph happens to satisfy these constraints, all subsequently generated graphs will be required to also satisfy them.

`reject_f4z`
: If this is a number greater than zero, all f4-statistics with `abs(z) > reject_f4z` will be used to constrain the search space of admixture graphs: Any graphs in which f4-statistics greater than `reject_f4z` are expected to be zero will not be evaluated.

`max_admix`
: Maximum number of admixture edges. By default, this number is equal to `numadmix`, or to the number of admixture edges in `initgraph`, so the number of admixture edges stays constant. Setting this to a higher number will lead to more admixture edges being added occasionally (see `plusminus_generations`). Graphs with additional admixture edges will only be accepted if they improve the score by 5% or more.

`verbose`
: Print progress updates

`...`
: Additional arguments passed to `qpgraph`

Value

A nested data frame with one model per line

See Also

`qpgraph`, `find_graphs_old`

Examples

```
## Not run:
```

```
res = find_graphs(example_f2_blocks, numadmix = 2)
res %>% slice_min(score)

## End(Not run)
## Not run:
# Start with a graph with 0 admixture events, increase up to 3,
# and stop after 10 generations of no improvement
pops = dimnames(example_f2_blocks)[[1]]
initgraph = random_admixturegraph(pops, 0, outpop = 'Chimp.REF')
res = find_graphs(example_f2_blocks, initgraph = initgraph, stop_gen2 = 10, max_admix = 3)
res %>% slice_min(score)

## End(Not run)
```

`find_graphs_old` *Find well fitting admixture graphs*

Description

This function generates and evaluates admixture graphs in `numgen` iterations across `numrep` independent repeats to find well fitting admixturegraphs. It uses the function `future_map` to parallelize across the independent repeats. The function `plan` can be called to specify the details of the parallelization. This can be used to parallelize across cores or across nodes on a compute cluster. Setting `numadmix` to 0 will search for well fitting trees, which is much faster than searching for admixture graphs with many admixture nodes.

Usage

```
find_graphs_old(
  data,
  pops = NULL,
  outpop = NULL,
  numrep = 1,
  numgraphs = 50,
  numgen = 5,
  numsel = 5,
  numadmix = 0,
  numstart = 1,
  keep = c("all", "best", "last"),
  initgraphs = NULL,
  mutfuns = namedList(spr_leaves, spr_all, swap_leaves, move_admixedge_once,
    flipadmix_random, mutate_n),
  mutprobs = NULL,
  opt_worst_residual = FALSE,
  store_intermediate = NULL,
  parallel = TRUE,
  stop_after = NULL,
  verbose = TRUE,
```

```
  ...
)
```

Arguments

`data`	Input data in one of three forms: 1. A 3d array of blocked f2 statistics, output of `f2_from_precomp` or `f2_from_geno` 2. A directory which contains pre-computed f2-statistics 3. The prefix of genotype files
`pops`	Populations for which to fit admixture graphs (default all)
`outpop`	An outgroup population which will split at the root from all other populations in all tested graphs. If one of the populations is know to be an outgroup, designating it as `outpop` will greatly reduce the search space compared to including it and not designating it as `outpop`.
`numrep`	Number of independent repetitions (each repetition can be run in parallel)
`numgraphs`	Number of graphs in each generation
`numgen`	Number of generations
`numsel`	Number of graphs which are selected in each generation. Should be less than `numgraphs`.
`numadmix`	Number of admixture events within each graph
`numstart`	Number of random initializations in each call to `qpgraph`. Defaults to 1, to speed up the graph optimization.
`keep`	Which models should be returned. One of `all`, `best`, `last` • `all` (default): Return all evaluated graphs • `best`: Return only the best fitting graph from each repeat and each generation • `last`: Return all graphs from the last generation
`initgraphs`	Optional graph or list of igraphs to start with. If `NULL`, optimization will start with random graphs.
`mutfuns`	Functions used to modify graphs. Defaults to the following: • `spr_leaves`: Subtree prune and regraft leaves. Cuts a leaf node and attaches it to a random other edge in the graph. • `spr_all`: Subtree prune and regraft. Cuts any edge and attaches the new orphan node to a random other edge in the graph, keeping the number of admixture nodes constant. • `swap_leaves`: Swaps two leaf nodes. • `move_admixedge_once`: Moves an admixture edge to a nearby location. • `flipadmix_random`: Flips the direction of an admixture edge (if possible). • `mutate_n`: Apply `n` of the mutation functions in this list to a graph (defaults to 2).

See examples for how to make new mutation functions.

`mutprobs` Relative frequencies of each mutation function.

- `NULL` (default) means each mutation function is picked with equal probability
- A numeric vector of length equal to `mutfuns` defines the relative frequency of each mutation function
- A matrix of dimensions `numgen` x `length(mutfuns)` defines the relative frequency of each mutation function in each generation

`opt_worst_residual`
Optimize for lowest worst residual instead of best score. `FALSE` by default, because the likelihood score is generally a better indicator of the quality of the model fit. Optimizing for the lowest worst residual is also slower (because f4-statistics need to be computed).

`store_intermediate`
Path and prefix of files for intermediate results to `.rds`. Can be useful if `find_graphs_old` doesn't finish sucessfully.

`parallel` Parallelize over repeats (if `numrep > 1`) or graphs (if `numrep == 1`) by replacing `map` with `future_map`. Will only be effective if `plan` has been set.

`stop_after` Stop optimization after `stop_after` seconds (and after finishing the current generation).

`verbose` Print progress updates

`...` Additional arguments passed to `qpgraph`

Value

A nested data frame with one model per line

See Also

`qpgraph`

Examples

```
## Not run:
find_graphs_old(example_f2_blocks, numrep = 200, numgraphs = 100,
            numgen = 20, numsel = 5, numadmix = 3)

## End(Not run)
## Not run:
# Making new mutation functions by modifying or combining existing ones:
newfun1 = function(graph, ...) mutate_n(graph, 3, ...)
newfun2 = function(graph, ...) flipadmix_random(spr_leaves(graph, ...), ...)
find_graphs_old(f2_blocks,
                mutfuns = namedList(spr_leaves, newfun1, newfun2),
                mutprobs = c(0.2, 0.3, 0.5))

## End(Not run)
```

find_newedges *Find possible new edges*

Description

Find possible new edges

Usage

```
find_newedges(graph, fix_outgroup = TRUE, all = TRUE)
```

Arguments

graph	An admixture graph
fix_outgroup	Exclude edges that would displace the root-to-outgroup edge (default TRUE).
all	If TRUE (default), use the fast algorithm; if FALSE, use the more conservative find_newedges_cautiously.

Value

A data frame with columns from and to. New edges which begin above from and end above to could be inserted

See Also

find_normedges find_admixedges

find_normedges *Find drift edges*

Description

Find drift edges

Usage

```
find_normedges(graph, exclude_first = FALSE)
```

Arguments

graph	An admixture graph
exclude_first	Do not return edge from root to outgroup

Value

A data frame with columns `from` and `to` with drift edges

See Also

`find_newedges` `find_admixedges`

---

flipadmix_random *Modify a graph flipping the direction of an admixture edge*

Description

Modify a graph flipping the direction of an admixture edge

Usage

```
flipadmix_random(graph, fix_outgroup = TRUE)
```

Arguments

`graph`	An admixture graph
`fix_outgroup`	Keep outgroup in place (has no effect here)

Value

A new admixture graph

---

fst *Compute Fst*

Description

This function computes pairwise Fst from genotype files, or from precomputed per-block Fst-statistics (not to be confused with per-block f2-statistics, which are used by most other functions). See `details` for how Fst is computed.

Usage

```
fst(data, pop1 = NULL, pop2 = NULL, boot = FALSE, verbose = FALSE, ...)
```

Arguments

`data`	Input data in one of three forms: 1. The prefix of genotype files 2. A directory with pre-computed Fst `.rds` files generated by `extract_f2` 3. A 3d array of blocked Fst, output of `f2_from_precomp` with `fst = TRUE`. This is not recommended, since pre-computed Fst blocks are not used by any other functions, and they can be mixed up with the more commonly used pre-computed f2 blocks.
`pop1`	One of the following four: 1. `NULL`: all possible population combinations will be returned 2. A vector of population labels. All combinations with the other `pop` arguments will be returned 3. A matrix with population combinations to be tested, with one population per column and one combination per row. Other `pop` arguments will be ignored. 4. the location of a file (`poplistname` or `popfilename`) which specifies the populations or population combinations to be tested. Other `pop` arguments will be ignored.
`pop2`	A vector of population labels
`boot`	If `FALSE` (the default), block-jackknife resampling will be used to compute standard errors. Otherwise, block-bootstrap resampling will be used to compute standard errors. If `boot` is an integer, that number will specify the number of bootstrap resamplings. If `boot = TRUE`, the number of bootstrap resamplings will be equal to the number of SNP blocks.
`verbose`	Print progress updates
`...`	Additional arguments passed to `f2_from_geno` when `data` is a genotype prefix

Details

The Hudson Fst estimator used here is described in the two publications below. For two populations with estimated allele frequency vectors `p1` and `p2`, and allele count vectors `n1` and `n2`, it is calculated as follows:

```
num = (p1 - p2)^2 - p1*(1-p1)/(n1-1) - p2*(1-p2)/(n2-1)
denom = p1 + p2 - 2*p1*p2
fst = mean(num)/mean(denom)
```

This is done independently for each SNP block, and is stored on disk for each population pair. Jackknifing or bootstrapping across these per-block estimates yields the overall estimates and standard errors.

References

Reich, D. (2009) *Reconstructing Indian population history* Nature

Bhatia, G. (2013) *Estimating and interpreting Fst: the impact of rare variants* Genome Research

Examples

```
## Not run:
pop1 = 'Denisova.DG'
pop2 = c('Altai_Neanderthal.DG', 'Vindija.DG')
fst(f2_dir, pop1, pop2)

## End(Not run)
```

`generate_all_graphs` *Generate all graphs*

Description

This functions generates all possible admixture graphs with a set number of admixture events for a given set of leaf nodes. It's pretty slow, and may not terminate in reasonable time for more than 5 leaves and 2 admixture events. The function is similar to the `all_graphs` function in the `admixturegraph` package, but there are a few differences:

- The function does not return graphs with fewer than `nadmix` admixture events
- The function does not return most graphs which are unidentifiable and would have equal fits as simpler identifiable graphs (for example it does not return graphs where a node is expanded to a loop)
- The function does not return duplicated graphs, as identified by the `graph_hash` function
- The function generates unique graphs which are missing in the output of `all_graphs`

Usage

```
generate_all_graphs(leaves, nadmix = 0, verbose = TRUE)
```

Arguments

`leaves`	The leaf nodes
`nadmix`	The number of admixture nodes
`verbose`	Print progress updates

Value

A list of graphs in `igraph` format

See Also

`all_graphs`, `generate_all_trees`, `graph_hash`

Examples

```
## Not run:
graphs = generate_all_graphs(letters[1:4], 1)

## End(Not run)
```

---

generate_all_trees *Generate all trees*

Description

This functions generates all possible trees with for a given set of leaf nodes.

Usage

```
generate_all_trees(leaves)
```

Arguments

leaves	The leaf nodes

Value

A list of trees in igraph format

---

get_block_lengths *Find LD-independent blocks*

Description

A new block begins at the SNP after the first SNP which is not within blgsize of the start of the last block. dat needs to be ordered first by 'CHR', then by 'POS' or 'cm'

Usage

```
get_block_lengths(dat, blgsize = 0.05, cpp = TRUE, verbose = TRUE)
```

Arguments

dat	Data frame with columns 'CHR' and either 'POS' or 'cm'
blgsize	SNP block size in Morgan. Default is 0.05 (5 cM). If blgsize is 100 or greater, if will be interpreted as base pair distance rather than centimorgan distance.
cpp	Should the faster C++ version be used?
verbose	Print progress updates

Value

A numeric vector where the ith element lists the number of SNPs in the ith block.

Examples

```
## Not run:
prefix = 'path/to/packedancestrymap_prefix'
pops = c('pop1', 'pop2', 'pop3')
afdat = packedancestrymap_to_afs(prefix, pops = pops)
block_lengths = get_block_lengths(afdat)

## End(Not run)
```

---

get_f2 *Turns f2_data into f2_blocks*

Description

Turns f2_data into f2_blocks

Usage

```
get_f2(f2_data, pops = NULL, pops2 = NULL, afprod = FALSE, verbose = TRUE, ...)
```

Arguments

f2_data	f2 data as genotype file prefix, f2 directory, or f2 blocks
pops	Populations for which to extract f2-stats (defaults to all)
pops2	Optional second vector of populations. Can result in non-square array
afprod	Get allele frequency products
verbose	Print progress updates
...	Additional arguments passed to f2_from_precomp or f2_from_geno

Value

A 3d array with f2-statistics

`get_leafnames` *Get the population names of a graph*

Description

Get the population names of a graph

Usage

```
get_leafnames(graph)
```

Arguments

`graph`	An admixture graph

Value

Population names

`get_outpop` *Get the outgroup from a graph (if it exists)*

Description

Get the outgroup from a graph (if it exists)

Usage

```
get_outpop(graph)
```

Arguments

`graph`	An admixture graph

Value

Outgroup name

`get_rootname` *Get the root name*

Description

Get the root name

Usage

```
get_rootname(graph)
```

Arguments

`graph`	An admixture graph

Value

Root name

`graph_addleaf` *Add a population to an admixture graph*

Description

Add a population to an admixture graph

Usage

```
graph_addleaf(graph, pop)
```

Arguments

`graph`	An admixture graph
`pop`	Population to add to the graph

Value

Admixture graph with the added population

See Also

`insert_leaf` adds pop at a specific position

graph_distances *Pairwise distance estimates for graphs*

Description

Computes a distance estimate for each graph pair. Each graph is first summarized as a vector which counts for every leaf pair how many internal nodes reach that pair. The distance between two graphs is the Euclidean distance between the vectors of two graphs, and is scaled to fall between 0 and 1.

Usage

```
graph_distances(graphlist)
```

Arguments

graphlist	List of graphs

Value

A data frame with graph distances

graph_equations *Find well fitting admixture graphs*

Description

This function generates and evaluates admixture graphs in `numgen` iterations to find well fitting admixturegraphs.

Usage

```
graph_equations(
  graph,
  substitute = TRUE,
  nam = c("a", "e", "f"),
  return_everything = FALSE
)
```

Arguments

graph	An admixture graph
substitute	Should edge names be represented by shorter symbols?
nam	Symbols used to shorten edge names
return_everything	If `TRUE`, return all intermediate columns in addition to equations; if `FALSE` (default), return only `equations` and `coding`.

Value

A list with two data frames: `equations` holds the equtions for all f2-statistics; `coding` has the mapping from edge names to edge symbols, which is used when `substitute = TRUE`

`graph_f2_function` *Make a function representing a graph*

Description

This function takes an igraph object and turns it into a function that takes edge weights as input, and outputs the expected f2-statistics.

Usage

```
graph_f2_function(
  graph,
  admix_default = 0.5,
  drift_default = 0.01,
  random_defaults = FALSE
)
```

Arguments

`graph`	An admixture graph
`admix_default`	The default weights for admixture edges
`drift_default`	The default weights for drift edges
`random_defaults`	Set default weights randomly for each edge between 0 and 1

Value

A function mapping edge weights to f2-statistics

Examples

```
## Not run:
mygraph = graph_f2_function(example_igraph)
mygraph(N3N8 = 0.1, `N2N1|Vindija.DG` = 0.4)

## End(Not run)
```

graph_flipadmix *Find all valid graphs which result from flipping one admixture edge*

Description

Find all valid graphs which result from flipping one admixture edge

Usage

```
graph_flipadmix(graph)
```

Arguments

graph	Admixture graph in igraph format

Value

A data frame with columns from, to, and graph

Examples

```
## Not run:
newgraphs = graph_flipadmix(example_igraph)
# now evaluate the new graphs
newgraphs %>%
  rowwise %>%
  mutate(res = list(qpgraph(example_f2_blocks, graph))) %>%
  unnest_wider(res)

## End(Not run)
```

graph_hash *Get unique hash of an admixture graph*

Description

This can be used to check if two graphs are identical. For two graphs to be identical, the topology and leaf nodes have to match, but internal node names do not matter.

Usage

```
graph_hash(graph)
```

Arguments

graph	Admixture graph in igraph format

Value

A hash string of the admixture graph

`graph_minusone` *Find all graphs which result from removing one admixture edge*

Description

Find all graphs which result from removing one admixture edge

Usage

```
graph_minusone(graph, ntry = Inf)
```

Arguments

`graph`	Admixture graph in `igraph` format
`ntry`	Maximum number of admixture edges to sample; if `Inf` (default), samples up to 100.

Value

A data frame with columns `from`, `to`, and `graph`

Examples

```
## Not run:
newgraphs = graph_minusone(example_igraph)
# now evaluate the new graphs
newgraphs %>%
  rowwise %>%
  mutate(res = list(qpgraph(example_f2_blocks, graph))) %>%
  unnest_wider(res)

## End(Not run)
```

`graph_minusplus` *Find all graphs which result from adding and removing one admixture edge*

Description

Find all graphs which result from adding and removing one admixture edge

Usage

```
graph_minusplus(graph)
```

Arguments

graph	Admixture graph in igraph format

Value

A data frame with columns source_from, source_to, dest_from, dest_to, and graph

Examples

```
## Not run:
newgraphs = graph_minusplus(example_igraph)
# now evaluate the new graphs
newgraphs %>%
  rowwise %>%
  mutate(res = list(qpgraph(example_f2_blocks, graph))) %>%
  unnest_wider(res)

## End(Not run)
```

---

graph_nodes *Return the nodes attribute of an edge tibble.*

Description

Returns the nodes tibble carried as an attribute on an edge tibble. Returns an empty zero-row tibble (with the canonical seven columns) when no attribute is attached. Errors on igraph or other non-data.frame inputs.

Usage

```
graph_nodes(graph)
```

Arguments

graph	An edge tibble.

Value

A tibble with columns name, samples, twoN_param, twoN, time_param, time, admix_event_time.

graph_plusone *Find all graphs which result from adding one admixture edge*

Description

Find all graphs which result from adding one admixture edge

Usage

```
graph_plusone(graph, ntry = Inf)
```

Arguments

graph	Admixture graph in igraph format
ntry	Specify this to return only a subset of all possible graphs with one more edge

Value

A data frame with columns from, to, and graph

Examples

```
## Not run:
newgraphs = graph_plusone(example_igraph)
# now evaluate the new graphs
newgraphs %>%
  rowwise %>%
  mutate(res = list(qpgraph(example_f2_blocks, graph))) %>%
  unnest_wider(res)

## End(Not run)
```

graph_splittrees *Find all trees which are part of the admixture graph*

Description

Find all trees which are part of the admixture graph

Usage

```
graph_splittrees(graph, return_admix = FALSE, simplify = TRUE)
```

Arguments

graph	Admixture graph in igraph format
return_admix	Include a column of admixture edges for each tree in the output (default FALSE).
simplify	Apply simplify_graph to each output tree (default TRUE).

Value

A data frame with columns name and graph

Examples

```
## Not run:
trees = graph_splittrees(example_igraph)
# now evaluate the trees
trees %>%
  rowwise %>%
  mutate(res = list(qpgraph(example_f2_blocks, graph))) %>%
  unnest_wider(res)

## End(Not run)
```

---

graph_to_afs *Simulate allele frequncies under an admixture graph*

Description

This function performs a very crude simulation of allele frequencies under an admixture graph model

Usage

```
graph_to_afs(
  graph,
  nsnps = 10000,
  drift_default = 0.02,
  admix_default = 0.5,
  leaves_only = FALSE
)
```

Arguments

graph	An admixture graph as igraph object, or as edge list data frame with a column weight, as returned by qpgraph()$edges
nsnps	Number of SNPs to simulate
drift_default	Default branch lengths. Ignored if graph is a data frame with weights
admix_default	Default admixture weights. Ignored if graph is a data frame with weights
leaves_only	Return allele frequencies for leaf nodes only

Value

A data frame with allele frequencies

See Also

graph_to_pcs

Examples

```
## Not run:
afs = graph_to_afs(example_igraph)

## End(Not run)
```

---

graph_to_lgo *Export an admixtools admixture graph to LEGOFIT .lgo format*

Description

Converts an admixtools admixture graph (as an edge tibble or igraph) into a LEGOFIT-compatible `.lgo` file.

Usage

```
graph_to_lgo(
  graph,
  file = NULL,
  samples = 1,
  dates_terminal = 0,
  outpop = NULL,
  twoN = NULL,
  time_handling = c("fix_admix", "init", "free"),
  drift_to_time = default_drift_to_time,
  fix_times = FALSE,
  validate = TRUE
)
```

Arguments

`graph`	An edge tibble (columns `from`, `to`, `type`, `weight`) or an igraph with `type` and `weight` edge attributes.
`file`	Output file path. If `NULL` (default), returns the `.lgo` text invisibly without writing.
`samples`	Samples per leaf. A scalar (applied to all leaves) or a named integer vector (per-leaf overrides). Default `1`.
`dates_terminal`	Starting time value for terminal (leaf) nodes. Default `0`.

`outpop` Name of an outgroup leaf to strip from the graph before export. `NULL` (default) skips stripping.

`twoN` Population size parameterization. `NULL` (default) emits `twoN fixed one=1` (coalescent units). A scalar numeric emits a fixed shared `twoN`. A named numeric vector emits per-segment free `twoN` parameters.

`time_handling` How to handle node times. One of:

- `"fix_admix"` (default): admixture-destination nodes get no `t=` declaration (their time is implicit). Requires drift values that are additively consistent along every root-to-leaf path.
- `"init"`: all non-leaf nodes are free starting points; respects an explicit `time` column on the edge tibble. Same consistency requirement as `"fix_admix"`.
- `"free"`: all non-leaf times are declared `free` with depth-based starting values. Bypasses the topology walk entirely, so it accepts edge tibbles whose drifts are not additively consistent (e.g., outputs from `qpgraph()`, which optimizes drift for f-stat fit rather than time-walk consistency).

`drift_to_time` Function converting per-edge drift to branch length. Default `default_drift_to_time()` (identity). Ignored when `time_handling = "free"` or when an explicit `time` column is present.

`fix_times` Logical (default `FALSE`). When `TRUE`, internal-node times are emitted as `time fixed` at their computed absolute values rather than `time free`. Use this with a free `twoN` (a named `twoN=` vector) when you need to recover **absolute** effective population sizes: with both times and `twoN` free, site-pattern data fit only the `Δt/twoN` ratios, so the absolute scale is unidentified (the data constrain the ratios but not the overall scale). Fixing the times removes that degeneracy. With `"fix_admix"` or `"init"` the fixed values are the real absolute times, so the recovered `twoN` is absolute. With `"free"` (the only mode that exports additively-inconsistent graphs, e.g. most admixture topologies) the fixed values are topological-depth placeholders: still consistent and still enough to make `twoN` identifiable in a self-consistent fit, but not biologically calibrated absolute times.

`validate` If `TRUE` (default), round-trips the output through `read_lgo()` to confirm topology is preserved.

Value

The `.lgo` text, invisibly.

Examples

```
# A 3-leaf no-admixture graph with explicit node times.
g <- tibble::tribble(
  ~from,  ~to,  ~type,    ~weight,  ~time,
  "xyz",  "xy", "normal", NA_real_, 1.5,
  "xyz",  "z",  "normal", NA_real_, 2,
  "xy",   "x",  "normal", NA_real_, 0.5,
  "xy",   "y",  "normal", NA_real_, 0.5
```

```
)
cat(graph_to_lgo(g, time_handling = "init"))
```

`graph_to_pcs` *Simulate PCs under an admixture graph*

Description

This function simulates PCA of allele frequencies under an admixture graph model

Usage

```
graph_to_pcs(
  graph,
  nsnps = 10000,
  drift_default = 0.02,
  admix_default = 0.5,
  leaves_only = TRUE
)
```

Arguments

`graph`	An admixture graph as igraph object, or as edge list data frame with a column `weight`, as returned by `qpgraph()$edges`
`nsnps`	Number of SNPs to simulate
`drift_default`	Default branch lengths. Ignored if `graph` is a data frame with weights
`admix_default`	Default admixture weights. Ignored if `graph` is a data frame with weights
`leaves_only`	Return PCs for leaf nodes only

Value

A data frame with PCs for each population

See Also

`graph_to_afs`

Examples

```
## Not run:
pcs = graph_to_pcs(example_igraph)
pcs %>% ggplot(aes(PC1, PC2, label = pop)) + geom_text() + geom_point()

## End(Not run)
```

`graph_to_qpadm` *Get all qpadm models for a graph*

Description

This function tests which qpadm models should be valid for an admixture graph and a target population. By default, all models returned by `qpadm_models` are tested. For large graphs this will be too slow, and you may want test only some models by providing the `models` argument, or only a single model by providing the `left` and `right` arguments.

Usage

```
graph_to_qpadm(
  graph,
  target,
  left = NULL,
  right = NULL,
  models = NULL,
  weights = TRUE,
  f4dat = NULL,
  allpops = TRUE,
  more_right = TRUE,
  return_f4 = FALSE,
  eps = 1e-10
)
```

Arguments

`graph`	An admixture graph
`target`	Name of the target population.
`left`	Left populations (provide this optionally if you want to test only a single qpadm model)
`right`	Right populations (provide this optionally if you want to test only a single qpadm model)
`models`	A two column nested data frame with models to be evaluated, one model per row. The first column, `l`, should contain the left populations, the second column, `r`, should contain the right populations. The target population is provided separately in the `target` argument.
`weights`	Set this to `FALSE` to return only information on the ranks, not the weights, of each qpadm model. The ranks should depend only on the graph topology, while the weights and weight-validity (all weights for left populations between 0 and 1) can depend on the branch lengths of the graph. By default f4-statistics are based on equal branch lengths and admixture weights of 0.5. This can be overridden by providing `f4dat`.
`f4dat`	A data frame of f4-statistics which can be provided to override the default branch lengths.

`allpops`	Evaluate only models which use all populations in the admixture graph. See `qpadm_models`
`more_right`	Passed to `qpadm_models`: prefer models with more right than left populations (default `TRUE`).
`return_f4`	Include f4 statistic matrices in the results (default `FALSE`)
`eps`	Epsilon value close to zero which is used for determining which f4 matrix elements should be considered non-zero, and which weights are strictly between 0 and 1.

Details

Two validity criteria are tested for each qpadm model: Rank validity and weight validity. Rank validity means that the rank of the f4 statistic matrix for only left and right populations is the same as the rank of the f4 statistic matrix that includes the target population among the left populations. Weight validity means that the estimated admixture weights for all left populations are between 0 and 1.

Value

A data frame with one qpadm model per row and columns `valid_rank` and `valid_weights` indicating whether a model should be valid under the graph.

Note

An earlier version of this function tried to use the graph topology for identifying valid qpadm models, but this didn't work reliably. Christian Huber had the idea of using the ranks of expected f4 statistic matrices instead.

See Also

`qpadm_models`, `graph_f2_function`

Examples

```
## Not run:
graph2 = example_igraph %>% simplify_graph() %>%
  delete_admix('N3N0', 'N3N4') %>% delete_admix('N3N1', 'N3N8')
graph_to_qpadm(graph2, 'Mbuti.DG') %>% filter(valid_rank, valid_weights)

## End(Not run)
```

`graphmod_pavel` *Return all graphs created from permuting a subclade*

Description

generates new graphs from basegraph as follows:

1. generates all possible trees using `addpops` (which are not in basegraph)
2. attaches trees to connection_edge, which is defined by two nodes in basegraph
3. adds edges originating above each edge in `source_node`, to each node above `addpops`

Usage

```
graphmod_pavel(basegraph, addpops, connection_edge, source_nodes)
```

Arguments

`basegraph`	an admixture graph as igraph object. (convert from edge list using `igraph::graph_from_edgelist`)
`addpops`	a vector of population labels which are not in `basegraph`. These populations should form a clade. All possible trees will be generated and those trees will be attached to `basegraph`.
`connection_edge`	edge in `basegraph` where the tree made from `addpops` should be attached
`source_nodes`	nodes in `basegraph`. edges above these nodes will be added and attached to all terminal edges leading to `addpops`

Examples

```
## Not run:
graphlist = graphmod_pavel(example_igraph, addpops = c('pop1', 'pop2', 'pop3'),
                           connection_edge = c('N2N0', 'N1N'),
                           source_nodes = c('Denisova.DG', 'N2N2'))
results = tibble(graph = graphlist) %>%
  mutate(res = map(graph, ~qpgraph(example_f2_blocks, .))) %>%
  unnest_wider(res) %>%
  mutate(worstz = map_dbl(f3, ~max(abs(.$z))))

## End(Not run)
```

`group_samples` *Group precomputed data*

Description

Computing f2 statistics for populations consisting of many individuals can be slow and require a lot of memory. To speed this up, this function groups individuals into populations, computes allele counts and products of pairwise allele counts with all individuals and groups, and writes the data to disk. f2 statistics for a combination of grouped and ungrouped precomputed data can then be read using `f2_from_precomp`, replacing individual IDs of the grouped samples with the new group labels. All groupings are listed in `{dir}/groups/{groupname}.rds`

Usage

```
group_samples(dir, inds, pops, overwrite = FALSE, verbose = TRUE)
```

Arguments

`dir`	Directory with precomputed individual pair data
`inds`	Individuals to group
`pops`	Group names, either length 1, or same length as `inds`
`overwrite`	Overwrite existing files in `outdir`
`verbose`	print progress updates

See Also

`delete_groups`

Examples

```
## Not run:
dir = 'my/f2/dir/'
inds = c('ind1', 'ind2', 'ind3', 'ind4', 'ind5')
pops = c('pop_A', 'pop_A', 'pop_A', 'pop_B', 'pop_B')
group_samples(dir, inds, pops)

## End(Not run)
```

igraph_to_agraph *Convert igraph to agraph*

Description

agraph is the format used by the admixturegraph package. igraph is used by the admixtools package

Usage

```
igraph_to_agraph(igraph)
```

Arguments

igraph	An admixture graph in igraph format (as used by admixtools).

Value

An admixture graph in agraph format (as used by the admixturegraph package).

Examples

```
## Not run:
igraph_to_agraph(example_igraph)

## End(Not run)
```

insert_admix *Insert a single edge into graph*

Description

Insert a single edge into graph

Usage

```
insert_admix(
  graph,
  source_from = NULL,
  source_to = NULL,
  dest_from = NULL,
  dest_to = NULL,
  substitute = FALSE,
  fix_outgroup = TRUE
)
```

Arguments

graph	An admixture graph
source_from	Parent node of the source edge
source_to	Child node of the source edge
dest_from	Parent node of the destination edge
dest_to	Child node of the destination edge
substitute	Should another edge be inserted, if the one specified doesn't work?
fix_outgroup	Prevent insertion of edges that affect the root-to-outgroup edge (default TRUE).

Value

Adxmiture graph with inserted edge

See Also

delete_admix, insert_admix_n

---

insert_admix_n *Insert admixture edges into graph*

Description

Insert admixture edges into graph

Usage

```
insert_admix_n(graph, n = 1, fix_outgroup = TRUE)
```

Arguments

graph	An admixture graph
n	Number of admixture edges to insert (default 1).
fix_outgroup	Prevent insertion of edges that affect the root-to-outgroup edge (default TRUE).

Value

Admixture graph with inserted edges

See Also

insert_admix delete_admix

insert_admix_old *Insert admixture edges into graph*

Description

Insert admixture edges into graph

Usage

```
insert_admix_old(
  graph,
  fromnodes,
  tonodes,
  substitute_missing = FALSE,
  allow_below_admix = FALSE
)
```

Arguments

graph	An admixture graph
fromnodes	List of nodes. New edges will originate above these nodes.
tonodes	List of nodes. New edges will end above these nodes.
substitute_missing	If TRUE, an attempt will be made to insert random other edges if some of the provided edges could not be inserted.
allow_below_admix	Allow insertion of edges which begin or end directly underneath an admixture node

Value

Adxmiture graph with inserted edges

insert_leaf *Add population to graph*

Description

Add population to graph

Usage

```
insert_leaf(graph, leaf, from, to)
```

Arguments

`graph`	An admixture graph
`leaf`	Population to be added
`from`	Source node of edge onto which `leaf` should be added
`to`	Target node of edge onto which `leaf` should be added

Value

Admixture graph with added population

See Also

`delete_leaf`, `graph_addleaf` to add `leaf` at any position

`is_valid` *Test if an admixture graph is valid*

Description

Test if an admixture graph is valid

Usage

```
is_valid(graph)
```

Arguments

`graph`	An admixture graph

Value

`TRUE` if graph is valid, otherwise `FALSE`

`isomorphism_classes` *Find identical graphs*

Description

Find identical graphs

Usage

```
isomorphism_classes(igraphlist)
```

Arguments

igraphlist	A list with admixture graphs

Value

An integer vector with isomorphism classes. Graphs with the same number have identical topology (but may have different labels).

See Also

isomorphism_classes2

---

isomorphism_classes2 *Find identical graphs*

Description

Find identical graphs

Usage

```
isomorphism_classes2(igraphlist)
```

Arguments

igraphlist	A list with admixture graphs

Value

An integer vector with isomorphism classes. Graphs with the same number have identical topology and leaf labels (but may have different internal labels).

See Also

isomorphism_classes

joint_sfs *Joint site frequency spectrum*

Description

This function computes the joint site frequency spectrum from genotype files or allele frequency and count matrices. The joint site frequency spectrum lists how often each combination of haplotypes is observed. The number of combinations is equal to the product of one plus the number of haplotypes in each population. For example, five populations with a single diploid individual each have 3^5 possible combinations.

Usage

```
joint_sfs(afs, pref = NULL)
```

Arguments

`afs`	A named list of length two where the first element (`afs`) contains the allele frequency matrix, and the second element (`counts`) contains the allele count matrix.
`pref`	Instead of `afs`, the prefix of genotype files can be provided.

Value

A data frame with the number of times each possible combination of allele counts is observed.

Examples

```
## Not run:
dat = plink_to_afs('/my/plink/file', pops = c('pop1', 'pop2', 'pop3', 'pop4', 'pop5'))
joint_sfs(dat)

## End(Not run)
```

joint_spectrum *Estimate joint allele frequency spectrum*

Description

Estimate joint allele frequency spectrum

Usage

```
joint_spectrum(afs)
```

Arguments

`afs`	A matrix or data frame of allele frequencies

Value

A data frame with columns `pattern` and `proportion`

Examples

```
## Not run:
dat = plink_to_afs('/my/plink/file', pops = c('pop1', 'pop2', 'pop3', 'pop4', 'pop5'))

# Spectrum across all SNPs
joint_spectrum(dat$afs)

# Stratify by allele frequency in one population
dat$afs %>% as_tibble %>% select(1:4) %>%
  group_by(grp = cut(pop1, 10)) %>%
  group_modify(joint_spectrum) %>%
  ungroup

# Stratify by mutation class
dat$afs %>% as_tibble %>% select(1:4) %>%
  mutate(mut = paste(dat$snpfile$A1, dat$snpfile$A2)) %>%
  group_by(mut) %>%
  group_modify(joint_spectrum) %>%
  ungroup

## End(Not run)
```

`lazadm` *Estimate admixture weights*

Description

Models target as a mixture of left populations, and outgroup right populations. Uses Lazaridis method based non-negative least squares of f4 matrix.

Usage

```
lazadm(data, left, right, target, boot = FALSE, constrained = TRUE)
```

Arguments

`data`	The input data in the form of: • A 3d array of blocked f2 statistics, output of `f2_from_precomp` or `extract_f2` • A directory with f2 statistics

- The prefix of a genotype file

`left`	Left populations (sources)
`right`	Right populations (outgroups)
`target`	Target population
`boot`	If `FALSE` (the default), block-jackknife resampling will be used to compute standard errors. Otherwise, block-bootstrap resampling will be used to compute standard errors. If `boot` is an integer, that number will specify the number of bootstrap resamplings. If `boot = TRUE`, the number of bootstrap resamplings will be equal to the number of SNP blocks.
`constrained`	Constrain admixture weights to be non-negative

Value

`lazadm` returns a data frame with weights and standard errors for each left population

References

Patterson, N. et al. (2012) *Ancient admixture in human history.* Genetics

Haak, W. et al. (2015) *Massive migration from the steppe was a source for Indo-European languages in Europe.* Nature (SI 9)

See Also

qpadm

Examples

```
target = 'Denisova.DG'
left = c('Altai_Neanderthal.DG', 'Vindija.DG')
right = c('Chimp.REF', 'Mbuti.DG', 'Russia_Ust_Ishim.DG', 'Switzerland_Bichon.SG')
lazadm(example_f2_blocks, left, right, target)
lazadm(example_f2_blocks, left, right, target, constrained = FALSE)
```

`loo_list` *Generate a list of leave-one-out arrays*

Description

Generate a list of leave-one-out arrays

Usage

```
loo_list(arr)
```

Arguments

`arr`	3d array with blocked estimates, with blocks in the 3rd dimension

Value

A list of leave-one-out arrays, each of which is 1 element shorter than `arr` along the 3rd dimension

See Also

`boo_list` `est_to_loo`

`loo_to_est` *Turn leave-one-out estimates to per-block estimates*

Description

Inverse of `est_to_loo` This works for any statistics which, when computed across `N` blocks, are equal to the weighted mean of the statistics across the `N` blocks.

Usage

```
loo_to_est(arr, block_lengths = NULL)
```

Arguments

`arr`	3d array with blocked estimates, with blocks in the 3rd dimension.
`block_lengths`	Optional block lengths. If `NULL`, will be parsed from 3rd dimnames in blocks

Value

A 3d array with leave-one-out estimates for jackknife. Dimensions are equal to those of `arr`.

See Also

`est_to_loo`

`move_admixedge_once` *Modify a graph by moving an admixture edge*

Description

Modify a graph by moving an admixture edge

Usage

```
move_admixedge_once(graph, fix_outgroup = TRUE)
```

Arguments

`graph`	An admixture graph
`fix_outgroup`	Keep outgroup in place

Value

A new admixture graph

`msprime_genome` *Simulate an admixture graph in msprime v1.x.*

Description

This function generates an msprime simulation script, and optionally executes it in python. Unlike `msprime_sim`, this function can simulate continuous sequence (not independent SNPs) and multiple chromosomes.

Usage

```
msprime_genome(
  graph,
  outpref = "msprime_sim",
  neff = 1000,
  ind_per_pop = 1,
  mutation_rate = 1.25e-08,
  time = 1000,
  fix_leaf = FALSE,
  nchr = 1,
  recomb_rate_chr = 2e-08,
  seq_length = 1000,
  admix_default = 0.5,
  run = FALSE,
  ghost_lineages = FALSE,
  shorten_admixed_leaves = FALSE
)
```

Arguments

graph	A graph as an `igraph` object or edge list with columns 'from' and 'to'. If it is an edge list with a column 'weight' (derived possibly from a fitted graph), the admixture weights will be used. Otherwise, all admixture edges will have a weight of 0.5.
outpref	A prefix of output files.
neff	Effective population size (in diploid individuals). If a scalar value, it will be constant across all populations. Alternatively, it can be a named vector with a different value for each population (e.g., `c('R'=100, 'A'=50, 'B'=50)`).
ind_per_pop	The number of diploid individuals to simulate for each population. If a scalar value, it will be constant across all populations. Alternatively, it can be a named vector with a different value for each population (e.g., `c('A'=10, 'B'=20)` to sample 10 and 20 diploid individuals from populations A and B, respectively).
mutation_rate	Mutation rate per site per generation. The default is `1.25e-8` per base pair per generation.
time	Either a scalar value (1000 generations by default) with the dates generated by `pseudo_dates`, or a named vector with dates for each graph node (in generations).
fix_leaf	A boolean value specifying if the dates of the leaf nodes will be fixed at time 0. If `TRUE`, all samples will be drawn at the end of the simulation (i.e., from "today").
nchr	Number of chromosomes to simulate.
recomb_rate_chr	A float value specifying recombination rate along the chromosomes. The default is `2e-8` per base pair per generation.
seq_length	The sequence length of the chromosomes. If it is a scalar value, the sequence length will be constant for all chromosomes. Alternatively, it can be a vector with a length equal to the number of chromosomes (i.e., `c(100,50)` to simulate 2 chromosomes with the lengths of 100 and 50 base pairs).
admix_default	A float value specifying default admixture proportion for all admixture nodes. The default is `0.5`. If another value between 0 and 1 is specified, admixture weights for each admixture event will be (value, 1-value).
run	If `FALSE`, the function will terminate after writing the msprime script. If `TRUE`, it will try to execute the msprime script with the default python installation. If you want to use some other python installation, you can set `run = /my/python`.
ghost_lineages	A boolean value specifying whether ghost lineages will be allowed. If `TRUE`, admixture happens at the time points defined by the y-axis generated while plotting the graph by `plot_graph`. If `FALSE` (default), admixture occurs at the time of the previous split event.

shorten_admixed_leaves
: If TRUE simulate the behavior of treemix where drift after admixture is not allowed

Value

The file name and path of the simulation script

Examples

```
## Not run:
results = qpgraph(example_f2_blocks, example_graph)
# Simulate 3 chromosomes whose lengths are 50, 100 and 100
msprime_genome(results$edges, nchr=3, seq_length=c(50, 100, 100))

## End(Not run)
```

---

msprime_sim *Simulate an admixture graph in msprime v1.x*

Description

This function generates an msprime simulation script, and optionally executes it in python. Unlike the msprime_genome function, this function simulates independent SNP sites.

Usage

```
msprime_sim(
  graph,
  outpref = "msprime_sim",
  nsnps = 1000,
  neff = 1000,
  ind_per_pop = 1,
  mutation_rate = 0.001,
  time = 1000,
  fix_leaf = FALSE,
  admix_default = 0.5,
  run = FALSE,
  numcores = NULL,
  ghost_lineages = FALSE,
  shorten_admixed_leaves = FALSE
)
```

Arguments

graph
: A graph as an igraph object or edge list with columns 'from' and 'to'. If it is an edge list with a column 'weight' (derived possibly from a fitted graph), the admixture weights will be used. Otherwise, all admixture edges will have a weight of 0.5.

`outpref` A prefix of output files.

`nsnps` The number of SNPs to simulate. All SNPs will be simulated independently of each other.

`neff` Effective population size (in diploid individuals). If a scalar value, it will be constant across all populations. Alternatively, it can be a named vector with a different value for each population (e.g., `c('R'=100, 'A'=50, 'B'=50)`).

`ind_per_pop` The number of diploid individuals to simulate for each population. If a scalar value, it will be constant across all populations. Alternatively, it can be a named vector with a different value for each population (e.g., `c('A'=10, 'B'=20)` to sample 10 and 20 diploid individuals from populations A and B, respectively).

`mutation_rate` Mutation rate per site per generation. The default is set to a high value (0.001 per site per generation) to obtain more polymorphic SNPs in order to speed up the simulation.

`time` Either a scalar value (1000 generations by default) with the dates generated by `pseudo_dates` function, or a named vector with dates for each graph nodes (in generations).

`fix_leaf` A boolean value specifying if the dates of the leaf nodes will be fixed at time 0. If `TRUE`, all samples will be drawn at the end of the simulation (i.e., from "today").

`admix_default` A float value specifying default admixture proportion for all admixture nodes. The default is `0.5`. If another value between 0 and 1 is specified, admixture weights for each admixture event will be (value, 1-value).

`run` If `FALSE`, the function will terminate after writing the msprime script. If `TRUE`, it will try to execute the msprime script with the default python installation. If you want to use some other python installation, you can set `run = /my/python`.

`numcores` The number of cores to use when simulating data.

`ghost_lineages`
A boolean value specifying whether ghost lineages will be allowed. If `TRUE`, admixture happens at the time points defined by the y-axis generated while plotting the graph by `plot_graph`. If `FALSE` (default), admixture occurs at the time of the previous split event.

`shorten_admixed_leaves`
If `TRUE` simulate the behavior of treemix where drift after admixture is not allowed

Value

The file name and path of the simulation script

Examples

```
## Not run:
results = qpgraph(example_f2_blocks, example_graph)
```

```
msprime_sim(results$edges, nsnps=100)

## End(Not run)
```

`mutate_n` *Modify a graph by applying n mutation functions*

Description

Modify a graph by applying n mutation functions

Usage

```
mutate_n(
  graph,
  n = 2,
  funs = list(spr_all, spr_leaves, swap_leaves, move_admixedge_once, flipadmix_random,
    mutate_n),
  fix_outgroup = TRUE
)
```

Arguments

`graph`	An admixture graph
`n`	Number of functions to apply
`funs`	List of function from which to choose
`fix_outgroup`	Keep outgroup in place

Value

A new admixture graph

`namedList` *Create a named list from arguments*

Description

Like `list()` but uses argument names as list names when names are not explicitly supplied.

Usage

```
namedList(...)
```

Arguments

`...`	Objects to include in the list.

Value

A named list.

`newick_to_edges` *Turn a newick format tree to a matrix of edges*

Description

Turn a newick format tree to a matrix of edges

Usage

```
newick_to_edges(newick, node = "R", edgemat = matrix(NA, 0, 2))
```

Arguments

`newick`	Tree in newick format.
`node`	Root label of the tree.
`edgemat`	Used for recursive function calls.

Value

Tree as two column matrix of edges (adjacency list)

Examples

```
newick = random_newick(c('a', 'b', 'c', 'd'))
newick
newick_to_edges(newick)
```

`node_counts` *Count how often each node in graph occurs in other graphs*

Description

Count how often each node in graph occurs in other graphs

Usage

```
node_counts(graph, graphlist)
```

Arguments

`graph`	An admixture graph
`graphlist`	List of graphs

`node_signature` *Returns a signature of a graph consisting of the left and right descendent leaf nodes of each internal node (sorted and concatenated)*

Description

Can be used to determine how often internal nodes occur in a list of other well fitting models

Usage

```
node_signature(graph)
```

Arguments

`graph`	An admixture graph

Value

A graph signature as character vector

Examples

```
## Not run:
sigs = example_winners %>% mutate(sig = map(igraph, node_signature)) %$%
         sig %>% unlist %>% table %>% c
node_signature(example_winners$igraph[[1]])

## End(Not run)
```

`node_times` *Get fitted times as a named numeric vector.*

Description

Backward-compatibility shim for callers that used the legacy `attr(result, "node_times")` attribute set by `read_legofit_output`. Resolves to `nodes$time` from the new nodes tibble.

Usage

```
node_times(graph)
```

Arguments

`graph`	An edge tibble.

Value

Named numeric vector keyed by node name.

`numadmix` *Count number of admixture nodes*

Description

Count number of admixture nodes

Usage

```
numadmix(graph)
```

Arguments

`graph`	An admixture graph

Value

Number of admixture nodes

`packedancestrymap_to_afs`
Read allele frequencies from packedancestrymap files

Description

Read allele frequencies from packedancestrymap files

Usage

```
packedancestrymap_to_afs(
  pref,
  inds = NULL,
  pops = NULL,
  adjust_pseudohaploid = TRUE,
  first = 1,
  last = NULL,
  verbose = TRUE
)
```

Arguments

`pref`	Prefix of packedancestrymap files (files have to end in `.geno`, `.ind`, `.snp`)
`inds`	Individuals from which to compute allele frequencies
`pops`	Populations from which to compute allele frequencies. If `NULL` (default), populations will be extracted from the third column in the `.ind` file. If population labels are provided, they should have the same length as `inds`, and will be matched to them by position
`adjust_pseudohaploid`	Genotypes of pseudohaploid samples are usually coded as `0` or `2`, even though only one allele is observed. `adjust_pseudohaploid` ensures that the observed allele count increases only by `1` for each pseudohaploid sample. If `TRUE` (default), samples that don't have any genotypes coded as `1` among the first 1000 SNPs are automatically identified as pseudohaploid. This leads to slightly more accurate estimates of f-statistics. Setting this parameter to `FALSE` is equivalent to the ADMIXTOOLS `inbreed: NO` option. Setting `adjust_pseudohaploid` to an integer `n` will check the first `n` SNPs instead of the first 1000 SNPs.
`first`	Index of the first SNP to read (1-based, default 1).
`last`	Index of the last SNP to read (1-based); if NULL, reads all SNPs.
`verbose`	Print progress updates

Value

A list with three data frames: allele frequency data, allele counts, and SNP metadata

Examples

```
## Not run:
afdat = packedancestrymap_to_afs(prefix, pops = pops)
afs = afdat$afs
counts = afdat$counts

## End(Not run)
```

---

packedancestrymap_to_plink

Convert EIGENSTRAT *or* PACKEDANCESTRYMAP *to* PLINK

Description

This function converts *EIGENSTRAT/PACKEDANCESTRYMAP* format files to *PLINK* files, using `write_plink`. When `inds` or `pops` is provided, only a subset of samples will be extracted. This function can have a high memory footprint, because data for all SNPs will be read before writing the *PLINK* files.

Usage

```
packedancestrymap_to_plink(
  inpref,
  outpref,
  inds = NULL,
  pops = NULL,
  verbose = TRUE
)
```

Arguments

inpref	Prefix of the input files
outpref	Prefix of the *PLINK* output files
inds	Individuals which should be extracted
pops	Populations which should be extracted. Can not be provided together with inds
verbose	Print progress updates

Alias

```
eigenstrat_to_plink
```

---

parse_dot *Read graph in dot format*

Description

Read graph in dot format

Usage

```
parse_dot(dotfile)
```

Arguments

dotfile	Name of a file with a dot formatted admixture graph

Examples

```
## Not run:
graph = parse_dot("/my/graph.dot")

## End(Not run)
```

`parse_fstats` *Parse qpGraph fstats output file*

Description

Parses the fstats output file written by the original qpGraph program.

Usage

```
parse_fstats(outfile, denom1 = 1000, denom2 = 1e+07)
```

Arguments

`outfile`	Path to the qpGraph fstats output file.
`denom1`	Scaling denominator for f3-statistics.
`denom2`	Scaling denominator for f4-statistics.

Value

A list of tibbles with parsed f-statistics.

`parse_qp3pop_output` *Read qp3Pop output file*

Description

Read qp3Pop output file

Usage

```
parse_qp3pop_output(outfile)
```

Arguments

`outfile`	output file generated by qp3Pop.

Value

tibble with output data.

`parse_qpadm_output` *Read qpAdm output file*

Description

Read qpAdm output file

Usage

```
parse_qpadm_output(outfile)
```

Arguments

`outfile`	Output file generated by qpAdm.

Value

Data frame with output data.

`parse_qpdstat_output` *Read qpDstat output file*

Description

Read qpDstat output file

Usage

```
parse_qpdstat_output(outfile)
```

Arguments

`outfile`	output file generated by qpDstat.

Value

tibble with output data.

parse_qpf4ratio_output

Read qpF4ratio output file

Description

Read qpF4ratio output file

Usage

```
parse_qpf4ratio_output(parfile)
```

Arguments

parfile	Output file generated by qpF4ratio

Value

Data frame with output data

parse_qpgraph_graphfile

Read qpGraph graph file

Description

Read qpGraph graph file

Usage

```
parse_qpgraph_graphfile(
  graphfile,
  split_multi = TRUE,
  igraph = FALSE,
  uselabels = FALSE
)
```

Arguments

graphfile	File with admixture graph in qpGraph format.
split_multi	Split multifurcations
igraph	Convert to igraph format
uselabels	Should labels or full names be used? Defaults to full names.

Value

Graph represented as two column edge matrix. Can have four columns if edges are locked

parse_qpgraph_output *Read qpGraph output file*

Description

Read qpGraph output file

Usage

```
parse_qpgraph_output(outfile, logfile = outfile)
```

Arguments

outfile	Output file generated by qpGraph.
logfile	Log file for reading score and outliers; defaults to outfile.

Value

list of output data.

permute_leaves *Modify a graph by permuting leaf nodes*

Description

Modify a graph by permuting leaf nodes

Usage

```
permute_leaves(graph, fix_outgroup = TRUE)
```

Arguments

graph	An admixture graph
fix_outgroup	Keep outgroup in place

Value

A new admixture graph

`pfile_to_afs` *Read allele frequencies from* `PFILE` *files (*`.pgen` */* `.pvar` */* `.psam`*)*

Description

Reads a PLINK 2 PFILE genotype triplet and returns the same shape as `plink_to_afs()` / `eigenstrat_to_afs()`: per-population allele frequencies and allele counts plus a SNP-metadata tibble. This unlocks the rest of the admixtools pipeline (`extract_f2`, `qpadm`, `qpgraph`, ...) on PFILE inputs without a PFILE -> BED conversion step.

Usage

```
pfile_to_afs(
  pref,
  inds = NULL,
  pops = NULL,
  adjust_pseudohaploid = TRUE,
  first = 1,
  last = NULL,
  numblocks = 1,
  poly_only = FALSE,
  multiallelic = c("error", "skip", "first_alt"),
  cm_file = NULL,
  verbose = TRUE
)
```

Arguments

`pref`	Prefix of PFILE files (the triplet `<pref>.pgen`, `<pref>.pvar`, `<pref>.psam`).
`inds`	Individuals from which to compute allele frequencies
`pops`	Populations from which to compute allele frequencies. If `NULL` (default), populations will be extracted from the third column in the `.ind` file. If population labels are provided, they should have the same length as `inds`, and will be matched to them by position
`adjust_pseudohaploid`	`TRUE` (default) probes the first 1000 variants and infers per-sample ploidy from observed-genotype uniqueness. Pass an integer to use that probe size instead. `FALSE` (or `0`) skips probing and assumes diploid for every sample.
`first`	Index of the first SNP to read (1-based, default 1).
`last`	Index of the last SNP to read (1-based); if NULL, reads all SNPs.
`numblocks`	Number of blocks in which to read genotype file. Setting this to a number greater than one is more memory efficient, but slower.
`poly_only`	Only keep SNPs with mean allele frequency not equal to 0 or 1 (default `FALSE`).

`multiallelic` Multiallelic-site policy: one of `"error"` (default), `"skip"`, or `"first_alt"`. See Details.

`cm_file` Optional path to a TSV companion file carrying per-variant centimorgan distances. PLINK 2 `.pvar` files do **not** carry genetic-map `cm` data in the standard format (they are VCF-derived; cm is not part of the VCF column spec). When `cm_file` is `NULL` and the `.pvar` has no `CM` column, `cm` defaults to `0` for all SNPs and a warning is issued; this matters because downstream `extract_f2(blgsize < 100)` uses `cm` for jackknife block boundaries; with `cm = 0` everywhere, every SNP collapses into a single block. The TSV must contain a SNP-id column named one of `{variant_id, SNP, ID, snp, id}` and a `cm` column. Extra columns are ignored. The file may have more rows than this prefix has SNPs; lookup is by SNP id and unmatched SNPs receive `NA`.

`verbose` Print progress updates

Details

The function is API-compatible with `plink_to_afs()` and returns an identically-shaped result, so any downstream code that consumes a `plink_to_afs` output works unchanged.

Reading the `.pgen` is delegated to the `pgenlibr` CRAN package (the canonical R binding to plink-ng's pgenlib, maintained by the plink2 author). No external plink2 binary is needed; everything runs in-process.

Format precedence in `anygeno_to_aftable()`:

When `anygeno_to_aftable()` auto-detects the format at a prefix and **both** PFILE (`.pgen` / `.pvar` / `.psam`) and BED (`.bed` / `.bim` / `.fam`) triplets exist, PFILE wins. PFILE is the newer of the two PLINK formats and the canonical interchange for plink2 pipelines, so a prefix that carries both is treated as PFILE-primary. To force BED dispatch in that situation, pass `format = "plink"` explicitly to `anygeno_to_aftable()` (or call `plink_to_afs()` directly).

Multiallelic sites:

admixtools' f-statistic model is biallelic. PFILE format supports multiallelic sites (ALT field with multiple comma-separated alleles). The `multiallelic` argument controls how those are handled:

- `"error"` (default): stop with an error listing a few example line numbers and suggesting a pre-filter command. Safest. Matches the assumption baked into the rest of admixtools.
- `"skip"`: drop multiallelic sites entirely from the output. Clean statistically: the dropped sites contribute nothing, the surviving biallelic sites are correct.
- `"first_alt"`: use the first ALT allele at each multiallelic site, ignoring the rest. **Produces biased f-statistics** unless the first ALT happens to be the major non-REF allele at every multiallelic site. PFILE allele ordering is determined by the input pipeline (e.g. `plink2 --make-pgen`'s alphabetical or first-encountered order) and is **not** guaranteed to put the major allele first. Use only if you have independently verified the ordering for your data, or accept the bias and document it. When in doubt prefer `"skip"`.

Value

A list with three items: allele-frequency matrix, allele-count matrix, and SNP-metadata tibble. Same shape as `plink_to_afs()`.

Examples

```
## Not run:
# PFILE without cm data: fine for analyses that don't need block jackknife
afdat = pfile_to_afs(prefix, pops = pops)

# PFILE with cm grafted from a separate genetic-map file
afdat = pfile_to_afs(prefix, pops = pops, cm_file = "panel_cm.tsv")

# Pre-filter multiallelic sites if necessary:
# system2("plink2", c("--pfile", prefix, "--max-alleles", "2",
#                     "--make-pgen", "--out", paste0(prefix, "_bi")))

## End(Not run)
```

`place_root_random` *Modify a graph by changing the position of the root node*

Description

Modify a graph by changing the position of the root node

Usage

```
place_root_random(graph, fix_outgroup = TRUE)
```

Arguments

`graph`	An admixture graph
`fix_outgroup`	Keep outgroup in place

Value

A new admixture graph

`plink_to_afs` *Read allele frequencies from* `PLINK` *files*

Description

Read allele frequencies from `PLINK` files

Usage

```
plink_to_afs(
  pref,
  inds = NULL,
  pops = NULL,
  adjust_pseudohaploid = TRUE,
  first = 1,
  last = NULL,
  numblocks = 1,
  poly_only = FALSE,
  verbose = TRUE
)
```

Arguments

`pref`	prefix of `PLINK` files (files have to end in `.bed`, `.bim`, `.fam`).
`inds`	Individuals from which to compute allele frequencies
`pops`	Populations from which to compute allele frequencies. If `NULL` (default), populations will be extracted from the third column in the `.ind` file. If population labels are provided, they should have the same length as `inds`, and will be matched to them by position
`adjust_pseudohaploid`	Genotypes of pseudohaploid samples are usually coded as `0` or `2`, even though only one allele is observed. `adjust_pseudohaploid` ensures that the observed allele count increases only by `1` for each pseudohaploid sample. If `TRUE` (default), samples that don't have any genotypes coded as `1` among the first 1000 SNPs are automatically identified as pseudohaploid. This leads to slightly more accurate estimates of f-statistics. Setting this parameter to `FALSE` is equivalent to the ADMIXTOOLS `inbreed: NO` option. Setting `adjust_pseudohaploid` to an integer `n` will check the first `n` SNPs instead of the first 1000 SNPs.
`first`	Index of the first SNP to read (1-based, default 1).
`last`	Index of the last SNP to read (1-based); if NULL, reads all SNPs.
`numblocks`	Number of blocks in which to read genotype file. Setting this to a number greater than one is more memory efficient, but slower.
`poly_only`	Only keep SNPs with mean allele frequency not equal to 0 or 1 (default `FALSE`).
`verbose`	Print progress updates

Value

A list with three items: Allele frequency matrix, allele count matrix, and SNP meta data.

Examples

```
## Not run:
afdat = plink_to_afs(prefix, pops)
afs = afdat$afs
counts = afdat$counts

## End(Not run)
```

---

plot_comparison *Compare two models*

Description

Plot a comparison of two runs of qpgraph or two runs of qpadm

Usage

```
plot_comparison(out1, out2, name1 = NULL, name2 = NULL)
```

Arguments

out1	First model
out2	Second model
name1	Optional name for first model
name2	Optional name for second model

Value

A ggplot object.

Examples

```
fit1 = qpgraph(example_f2_blocks, example_graph, lsqmode = FALSE)
fit2 = qpgraph(example_f2_blocks, example_graph, lsqmode = TRUE)
plot_comparison(fit1, fit2)
```

`plot_graph` *Plot an admixture graph*

Description

Plot an admixture graph

Usage

```
plot_graph(
  graph,
  fix = NULL,
  title = "",
  color = TRUE,
  textsize = 2.5,
  highlight_unidentifiable = FALSE,
  pos = NULL,
  dates = NULL,
  neff = NULL,
  scale_y = FALSE,
  hide_weights = FALSE
)
```

Arguments

`graph`	An admixture graph. If it's an edge list with a `label` column, those values will be displayed on the edges
`fix`	If `TRUE`, there will be an attempt to rearrange the nodes to minimize the number of intersecting edges. This can take very long for large graphs. By default this is only done for graphs with fewer than 10 leaves.
`title`	A plot title
`color`	Plot it in color or greyscale
`textsize`	Size of edge and node labels
`highlight_unidentifiable`	Highlight unidentifiable edges in red. Can be slow for large graphs. See `unidentifiable_edges`.
`pos`	An optional data frame with node coordinates (columns `node`, `x`, `y`)
`dates`	An optional named vector with dates (in generations) for each node to plot dates on the y-axis (e.g., `c('R'=1000, 'A'=0, 'B'=0)`). If this option is supplied, the y-axis will display dates in generations.
`neff`	An optional named vector with effective population sizes for each population (e.g., `c('R'=100, 'A'=100, 'B'=100)`). If this option is supplied, the effective population size of each population will be shown next to the corresponding edge.
`scale_y`	If `TRUE`, scale the y-axis according to `dates` vector. The default is `FALSE`.

`hide_weights` A boolean value specifying if the drift values on the edges will be hidden. The default is `FALSE`.

Value

A ggplot object

Examples

```
## Not run:
plot_graph(example_graph)

# Plot a random simulation output. Show dates and population sizes on the plot
out = random_sim(nleaf=5, nadmix=1)
plot_graph(out$edges, dates=out$dates, neff=out$neff)

## End(Not run)
```

---

plot_graph_map *Plot an admixture graph on a map*

Description

Plot an admixture graph on a map

Usage

```
plot_graph_map(graph, leafcoords, shapedata = NULL)
```

Arguments

`graph`	a two column matrix specifying the admixture graph. first column is source node, second column is target node. the first edge has to be root -> outgroup. admixture nodes are inferred as those nodes which are the targets of two different sources.
`leafcoords`	data frame with columns `group`, `lon`, `lat`
`shapedata`	shapedata

Value

a plotly object

Examples

```
## Not run:
plot_graph_map(example_igraph, example_anno)

## End(Not run)
```

plot_map *Plot samples on a map*

Description

Plot samples on a map

Usage

```
plot_map(
  leafcoords,
  map_layout = 1,
  color = "yearsbp",
  colorscale = "Portland",
  collog10 = TRUE
)
```

Arguments

leafcoords	data frame with columns iid, lon, lat
map_layout	1 or 2
color	color
colorscale	colorscale
collog10	collog10

Value

a plotly object.

Examples

```
## Not run:
plot_map(example_anno, 1)

## End(Not run)
```

plotly_comparison *Compare two models*

Description

Plot a comparison of two runs of qpgraph or two runs of qpadm

Usage

```
plotly_comparison(out1, out2, name1 = NULL, name2 = NULL)
```

Arguments

`out1`	First model
`out2`	Second model
`name1`	Optional name for first model
`name2`	Optional name for second model

Value

A plotly object.

Examples

```
## Not run:
fit1 = qpgraph(example_f2_blocks, example_graph, lsqmode = FALSE)
fit2 = qpgraph(example_f2_blocks, example_graph, lsqmode = TRUE)
plotly_comparison(fit1, fit2)

## End(Not run)
```

`plotly_graph` *Plot an admixture graph using plotly*

Description

Plot an admixture graph using plotly

Usage

```
plotly_graph(
  graph,
  collapse_threshold = 0,
  fix = FALSE,
  print_highlow = FALSE,
  highlight_unidentifiable = FALSE,
  pos = NULL,
  nudge_y = -0.1,
  annot = ""
)
```

Arguments

`graph`	An admixture graph
`collapse_threshold`	Collapse nodes if they are separated by less than this distance (for fitted graphs)

`fix`	If `TRUE`, there will be an attempt to rearrange the nodes to minimize the number of intersecting edges. This can take very long for large graphs. By default this is only done for graphs with fewer than 10 leaves.
`print_highlow`	Print high/low weight labels on admixture edges (default `FALSE`).
`highlight_unidentifiable`	Highlight unidentifiable edges in red. Can be slow for large graphs. See `unidentifiable_edges`.
`pos`	Optional data frame with node coordinates (columns `node`, `x`, `y`)
`nudge_y`	Vertical offset applied to node labels (default `-0.1`).
`annot`	Annotation string appended to the graph (default empty string).

Value

A plotly object

Examples

```
## Not run:
plotly_graph(example_graph)

## End(Not run)
```

---

pseudo_dates *Get pseudo dates for graph nodes*

Description

This function assigns a date (in generations) to each node in an admixture graph and is used in `msprime_sim` and `msprime_genome`. The date of the node will correspond to the y-coordinate of that node used for plotting in `plotly_graph`, unless the `fix_leaf` option is set to `TRUE`, in which case the dates of all leaf nodes returned by `get_leafnames` will be set to 0.

Usage

```
pseudo_dates(graph, time = 1000, fix_leaf = FALSE)
```

Arguments

`graph`	An admixture graph
`time`	A scalar by which y-coordinate values will be multiplied to get dates
`fix_leaf`	A boolean specifying if the dates of the leaf nodes will be fixed at time 0 (i.e., at the most recent time). If `TRUE`, all samples will be drawn at the end of the simulation (i.e., from "today"). The default is `FALSE`

Value

A named vector with pseudo dates for each graph node

`qp3pop` *Estimate f3 statistics*

Description

Computes f3 statistics of the form $f3(A; B, C)$. This is generally equivalent to $(f2(A, B) + f2(A, C) - f2(B, C))/2$ and to $f4(A, B; A, C)$. However, the exact estimates depend on the type of input data, selection of SNPs, and on the values of some of the arguments, which are described below.

Usage

```
qp3pop(
  data,
  pop1 = NULL,
  pop2 = NULL,
  pop3 = NULL,
  boot = FALSE,
  sure = FALSE,
  unique_only = TRUE,
  blgsize = NULL,
  block_lengths = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`data`	Input data in one of three forms: 1. A 3d array of blocked f2 statistics, output of `f2_from_precomp` or `f2_from_geno` (fastest option) 2. A directory which contains pre-computed f2-statistics 3. The prefix of genotype files (slowest option)
`pop1`	One of the following four: 1. `NULL`: all possible population combinations will be returned 2. A vector of population labels. All combinations with the other `pop` arguments will be returned 3. A matrix with population combinations to be tested, with one population per column and one combination per row. Other `pop` arguments will be ignored. 4. the location of a file (`poplistname` or `popfilename`) which specifies the populations or population combinations to be tested. Other `pop` arguments will be ignored.
`pop2`	A vector of population labels
`pop3`	A vector of population labels

boot	If FALSE (the default), block-jackknife resampling will be used to compute standard errors. Otherwise, block-bootstrap resampling will be used to compute standard errors. If boot is an integer, that number will specify the number of bootstrap resamplings. If boot = TRUE, the number of bootstrap resamplings will be equal to the number of SNP blocks.
sure	The number of population combinations can get very large. This is a safety option that stops you from accidently computing all combinations if that number is large.
unique_only	If TRUE (the default), redundant combinations will be excluded
blgsize	SNP block size in Morgan when computing from genotype data (default 0.05).
block_lengths	Optional vector of block lengths; if NULL, inferred from the data.
verbose	Print progress updates
...	Additional arguments passed to f3blockdat_from_geno if data is a genotype prefix, or to get_f2 otherwise. Commonly passed arguments include apply_corr, outgroupmode, and poly_only.

Details

The default values of the parameters apply_corr, outgroupmode, poly_only depend on the first argument (data), and they can affect the estimated f3-statistics. When the data is the prefix of genotype data, the default parameters are generally the same as in the original qp3pop program. When data is an array of pre-computed f2-statistics, the parameters may be set while computing f2-statistics. If the first population is a single pseudodiploid sample, it is not possible to get unbiased estimates of f3. To get biased estimates for a single pseudodiploid sample from genotype data, set outgroupmode = TRUE and apply_corr = FALSE. Under this combination of parameters, f3(A; B, C) should also be identical to f4(A, B; A, C), since f4-statistics generally do not require any bias correction. See examples for more information.

Value

qp3pop returns a data frame with f3 statistics, with columns for populations 1 to 3, f3-estimate (est), standard error of the estimate (se), z-score (z, est/se), p-value (2*(1-pnorm(z))), and the number of SNPs used (n; only if first argument is genotype prefix)

Alias

f3

References

Patterson, N. et al. (2012) *Ancient admixture in human history* Genetics

Peter, B. (2016) *Admixture, Population Structure, and F-Statistics* Genetics

Examples

```
## Not run:
pop1 = 'Denisova.DG'
pop2 = c('Altai_Neanderthal.DG', 'Vindija.DG')
pop3 = c('Chimp.REF', 'Mbuti.DG', 'Russia_Ust_Ishim.DG')
qp3pop(example_f2_blocks, pop1, pop2, pop3)

pops = c('Chimp.REF', 'Mbuti.DG', 'Russia_Ust_Ishim.DG')
qp3pop(example_f2_blocks, pops)
qp3pop(example_f2_blocks, pops, unique_only = FALSE)
qp3pop(f2_dir, pop1, pop2, pop3)

# Below are three scenarios, and in each one `qp3pop()` and `qp3pop_wrapper()`
# should give the same or very similar estimates. Note that to compute `f3(A; B, C)`,
# `qp3pop_wrapper()`, and the original qp3pop program, expect populations
# to be in the order `B`, `C`, `A`.

prefix = '/path/to/geno/prefix'
qp3popbin = '/path/to/AdmixTools/bin/qp3Pop'
pops = dimnames(example_f2_blocks)[[1]]

# 1. target diploid, outgroupmode NO (this is the default when passing a geno file prefix)
qp3pop_wrapper(prefix, pops[2], pops[3], pops[1], bin = qp3popbin, outgroupmode = FALSE)
qp3pop(prefix, pops[1], pops[2], pops[3])
  # est, se, z match well; n is higher
qp3pop(prefix, pops[1], pops[2], pops[3], poly_only = TRUE)
  # est, se, z match less well; n is identical

# 2. target diploid, outgroupmode YES (this is the only option with precomputed f2-stats)
qp3pop_wrapper(prefix, pops[2], pops[3], pops[1], bin = qp3popbin, outgroupmode = TRUE)
qp3pop(prefix, pops[1], pops[2], pops[3], outgroupmode = TRUE)
  # est, se, z match (except for factor 1000)
f2b = f2_from_geno(prefix, pops = pops[1:3], poly_only = FALSE)
qp3pop(f2b, pops[1], pops[2], pops[3])

# 3. target pseudodiploid (no heterozygotes means heterozygosity rate correction is not possible)
qp3pop_wrapper(prefix, pops[1], pops[3], pops[2], bin = qp3popbin, outgroupmode = TRUE)
qp3pop(prefix, pops[2], pops[1], pops[3], outgroupmode = TRUE, apply_corr = FALSE)
  # est, se, z match (except for factor 1000)

## End(Not run)
```

`qp3pop_wrapper` *Wrapper function around the original qp3Pop program*

Description

Computes f3 statistics of the form $f3(A; B, C)$. Equivalent to $(f2(A, B) + f2(A, C) - f2(B, C))/2$ and to $f4(A, B; A, C)$. Requires a working installation of qp3Pop, which will be called using `system`

Usage

```
qp3pop_wrapper(
  pref,
  source1,
  source2 = NULL,
  target = NULL,
  bin = "~np29/o2bin/qp3Pop",
  outdir = ".",
  parfile = NULL,
  inbreed = "NO",
  outgroupmode = "NO",
  printonly = FALSE,
  env = "",
  verbose = TRUE
)
```

Arguments

`pref`	Path to and prefix of the packedancestrymap genotype files
`source1`	One of the following four: 1. `NULL`: Populations will be read from `poplistname` or `popfilename` specified in `parfile` 2. A vector of population labels 3. A data frame in which the first four columns specify the population triples to be tested. `source2`, `target` will be ignored. 4. The location of a file (`poplistname` or `popfilename`) which specifies the populations or population combinations to be tested. `source2` and `target` will be ignored.
`source2`	A vector of population labels
`target`	A vector of population labels
`bin`	Path to the qp3Pop binary file
`outdir`	Output directory. files `out`, `parfile`, `poplistname`, `popfilename` may be overwritten
`parfile`	qp3Pop parameter file. If this is specified, `source1`, `source2`, `target` will be ignored.
`inbreed`	inbreed
`outgroupmode`	outgroupmode
`printonly`	Should the command be printed or executed?
`env`	Export environmental variables. See examples.
`verbose`	Print progress updates

Value

If `printonly`, the `qp3Pop` command, otherwise a data frame with parsed `qp3Pop` output

Examples

```
## Not run:
source1 = c('Altai_Neanderthal.DG', 'Vindija.DG')
source2 = c('Chimp.REF', 'Mbuti.DG', 'Russia_Ust_Ishim.DG')
target = 'Denisova.DG'
qp3pop_wrapper('genotype_prefix', source1, source2, target,
  bin = 'path/to/qp3Pop')

## End(Not run)
```

---

qpadm *Estimate admixture weights*

Description

qpadm models a target population as a mixture of left (source) populations, given a set of right (outgroup) populations. It can be used to estimate whether the left populations explain all genetic variation in the target population, relative to the right populations, and to estimate admixture proportions of the left populations to the target population.

Usage

```
qpadm(
  data,
  left,
  right,
  target,
  f4blocks = NULL,
  fudge = 1e-04,
  fudge_twice = FALSE,
  auto_only = TRUE,
  blgsize = 0.05,
  poly_only = FALSE,
  boot = FALSE,
  getcov = TRUE,
  constrained = FALSE,
  return_f4 = FALSE,
  cpp = TRUE,
  singular_threshold = NA_real_,
  verbose = TRUE,
  ...
)
```

Arguments

data	The input data in the form of: • A 3d array of blocked f2 statistics, output of f2_from_precomp or extract_f2

- A directory with f2 statistics
- The prefix of a genotype file

`left` Left populations (sources)

`right` Right populations (outgroups)

`target` Target population

`f4blocks` Instead of f2 blocks, f4 blocks can be supplied. This is used by `qpadm_multi`

`fudge` Value added to diagonal matrix elements before inverting

`fudge_twice` Setting this to `TRUE` should result in p-values that better match those in the original qpAdm program

`auto_only` Use only chromosomes 1 to 22.

`blgsize` SNP block size in Morgan. Default is 0.05 (5 cM). If `blgsize` is 100 or greater, if will be interpreted as base pair distance rather than centimorgan distance.

`poly_only` Exclude sites with identical allele frequencies in all populations.

`boot` If `FALSE` (the default), block-jackknife resampling will be used to compute standard errors. Otherwise, block-bootstrap resampling will be used to compute standard errors. If `boot` is an integer, that number will specify the number of bootstrap resamplings. If `boot = TRUE`, the number of bootstrap resamplings will be equal to the number of SNP blocks.

`getcov` Compute weights covariance. Setting `getcov = FALSE` will speed up the computation.

`constrained` Constrain admixture weights to be non-negative

`return_f4` Return f4-statistics

`cpp` Use C++ functions. Setting this to `FALSE` will be slower but can help with debugging.

`singular_threshold`
If not `NA` (the default), error out when the reciprocal condition number of the (fudged) f4 variance matrix is below this threshold. A common choice is `1e-12` – below that the downstream `solve()` and the C++ weight optimization fall back to a pseudo-inverse, which makes the returned weight standard errors numerically tight in a way that doesn't reflect actual sampling uncertainty. The default (`NA`) preserves the historical behavior: silently fall through to the pseudo-inverse fallback. In either mode, the reciprocal condition number is reported as `f4_var_rcond` on the returned list so callers can gate downstream interpretations programmatically.

Low-level RcppArmadillo `solve(): system is singular; attempting approx solution` stderr lines are suppressed at compile time via `ARMA_WARN_LEVEL 1`. Those lines previously flooded stderr on batch fits at higher K without carrying any information not already exposed by the R-side diagnostics. The canonical programmatic signal is `f4_var_rcond` (always populated, regardless of `verbose`); `qpadm_multi()` and `qpadm_sweep()` force `verbose = FALSE` internally, so the interactive `warning()` only fires for direct `qpadm()` calls at `verbose = TRUE`. Batch callers should inspect `f4_var_rcond` (and `f4_var_singular_loadings`) on the returned object rather than rely on stderr / warning output.

`verbose`	Print progress updates
`...`	If `data` is the prefix of genotype files, additional arguments will be passed to `f4blockdat_from_geno`

Value

`qpadm` returns a list with up to four data frames describing the model fit, plus a numeric `f4_var_rcond` diagnostic:

1. `f4_var_rcond`: Reciprocal condition number of the (fudged) f4 variance matrix that was inverted to compute weights and standard errors. Values near machine epsilon (~`1e-15`) indicate that the right populations are linearly dependent (e.g., a right pop is a sister to one of the sources), and the returned weight SEs may be artificially tight as a result of the pseudo-inverse fallback. See `singular_threshold` for opt-in error gating.
2. `f4_var_singular_loadings`: When `f4_var_rcond` is concerningly low ($<$ `1e-8`), a tibble with one row per right population (excluding the reference pop `right[1]`), sorted by each right pop's leverage on the collinear (near-singular) subspace. The leverage isolates right-side collinearity (it is derived from the right pops' own resampled f4 estimates, so a left-side singularity does not inflate it) and is the L2 norm of that right pop's rows of the orthogonal projector onto the degenerate subspace (the square root of the projector's diagonal), which is basis-invariant and therefore reproducible across linear-algebra (LAPACK/BLAS) backends. Right pops at the top of this list are the most likely offenders – typically a sister-clade pair among the right set (two pops with similar high loadings); loadings that are all near zero indicate the near-singularity originates on the left (the sources) rather than among the right populations. `NULL` otherwise.
3. `weights`: A data frame with estimated admixture proportions where each row is a left population.
4. `f4`: A data frame with estimated and fitted f4-statistics
5. `rankdrop`: A data frame describing model fits with different ranks, including p-values for the overall fit and for nested models (comparing two models with rank difference of one). A model with `L` left populations and `R` right populations has an f4-matrix of dimensions `(L-1)*(R-1)`. If no two left population form a clade with respect to all right populations, this model will have rank `(L-1)*(R-1)`.
   - `f4rank`: Tested rank
   - `dof`: Degrees of freedom of the chi-squared null distribution: `(L-1-f4rank)*(R-1-f4rank)`
   - `chisq`: Chi-sqaured statistic, obtained as `E'QE`, where `E` is the difference between estimated and fitted f4-statistics, and `Q` is the f4-statistic covariance matrix.
   - `p`: p-value obtained from `chisq` as `pchisq(chisq, df = dof, lower.tail = FALSE)`
   - `dofdiff`: Difference in degrees of freedom between this model and the model with one less rank
   - `chisqdiff`: Difference in chi-squared statistics
   - `p_nested`: p-value testing whether the difference between two models of rank difference 1 is significant
6. `popdrop`: A data frame describing model fits with different populations. Note that all models with fewer populations use the same set of SNPs as the first model.

- `pat`: A binary code indicating which populations are present in this model. A `1` represents dropped populations. The full model is all zeros.
- `wt`: Number of populations dropped
- `dof`: Degrees of freedom of the chi-squared null distribution: `(L-1-f4rank)*(R-1-f4rank)`
- `chisq`: Chi-sqaured statistic, obtained as `E'QE`, where `E` is the difference between estimated and fitted f4-statistics, and `Q` is the f4-statistic covariance matrix.
- `p`: *p*-value obtained from `chisq` as `pchisq(chisq, df = dof, lower.tail = FALSE)`
- `f4rank`: Tested rank
- `feasible`: A model is feasible if all weights fall between 0 and 1
- `<population name>`: The weights for each population in this model

References

Haak, W. et al. (2015) *Massive migration from the steppe was a source for Indo-European languages in Europe.* Nature (SI 10)

See Also

qpwave, lazadm

Examples

```
left = c('Altai_Neanderthal.DG', 'Vindija.DG')
right = c('Chimp.REF', 'Mbuti.DG', 'Russia_Ust_Ishim.DG', 'Switzerland_Bichon.SG')
target = 'Denisova.DG'
qpadm(example_f2_blocks, left, right, target)
## Not run:
# The original ADMIXTOOLS qpAadm program has an option called "allsnps"
# that selects different SNPs for each f4-statistic, which is
# useful when working with sparse genotype data.
# To get the same behavior in ADMIXTOOLS 2, supply the genotype data prefix
# and set `allsnps = TRUE`
qpadm("/my/geno/prefix", left, right, target, allsnps = TRUE)

## End(Not run)
```

---

qpadm_models *Partition a list of populations into left and right populations*

Description

Partition a list of populations into left and right populations

Usage

```
qpadm_models(pops, allpops = TRUE, more_right = TRUE)
```

Arguments

pops	A vector of populations
allpops	Return only models which use all provided populations
more_right	Return only models with more right than left populations

Value

A data frame with one qpadm model per row

See Also

qpadm_models, graph_f2_function

Examples

```
## Not run:
graph_to_qpadm(get_leafnames(example_igraph))

## End(Not run)
```

---

qpadm_models_old *Return all valid qpAdm models for an admixturegraph*

Description

For large admixturegraph, there may be a large number of valid qpAdm models!

Usage

```
qpadm_models_old(graph, add_outgroup = FALSE, nested = TRUE, abbr = -1)
```

Arguments

graph	An admixture graph as igraph object
add_outgroup	Should the graph outgroup be added to the qpAdm right populations? Technically this shouldn't be an informative outgroup for qpAdm.
nested	Should a nested data frame be returned (TRUE), or should populations be concatenated into strings (FALSE)?
abbr	Maximum number of characters to print for each population. The default (-1) doesn't abbreviate the names.

Examples

```
## Not run:
qpadm_models_old(igraph2, add_outgroup = TRUE)
qpadm_models_old(igraph2, add_outgroup = TRUE) %>% slice(1) %$% list(target, left, right)

## End(Not run)
```

`qpadm_multi` *Run multiple qpadm models*

Description

This function runs multiple qpadm models, re-using f4-statistics where possible. Supports parallel evaluation of models, which can be turned on with `future::plan('multisession')` or similar, and turned off with `future::plan('sequential')`.

Usage

```
qpadm_multi(
  data,
  models,
  allsnps = FALSE,
  full_results = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`data`	The input data in the form of: • A 3d array of blocked f2 statistics, output of `f2_from_precomp` or `extract_f2` • A directory with f2 statistics • The prefix of a genotype file
`models`	A nested list (or data frame) with qpadm models. It should consist of two or three other named lists (or columns) containing `left`, `right`, (and `target`) populations.
`allsnps`	Use all SNPs with allele frequency estimates in every population of any given population quadruple. If `FALSE` (the default) only SNPs which are present in all populations in `popcombs` (or any given model in it) will be used. When there are populations with lots of (non-randomly) missing data, `allsnps = TRUE` can lead to false positive results. This option only has an effect if `data` is the prefix of a genotype file. If `data` are f2-statistics, the behavior will be determined by the options that were used in computing the f2-statistics.
`full_results`	Return the full qpadm output list for each model; if `FALSE`, return only summary statistics (default `TRUE`). Note: the lean (`FALSE`) path dispatches to `qpadm_p()`, which omits `f4_var_rcond` and `f4_var_singular_loadings` from each per-fit result.
`verbose`	Print progress updates

... Further arguments forwarded to downstream functions. When `data` is a genotype-file prefix, kwargs matching `f4blockdat_from_geno()`'s formals are routed to the preflight f4 computation. Kwargs matching the per-fit function's formals (`qpadm` when `full_results=TRUE`, `qpadm_p()` when `FALSE`) are routed to each per-model fit. Unrecognised names raise a single error at entry naming both surfaces. `singular_threshold` (a `qpadm()` formal) is forwarded but be aware that if any model trips its threshold, `qpadm()` raises an error, the entire `qpadm_multi` call aborts via `furrr::future_map`, and no results are returned. For "mark and continue" semantics over a batch, omit `singular_threshold` and filter on `f4_var_rcond` after the call.

Value

A list where each element is the output of one qpadm model.

Examples

```
## Not run:
# the following specifies two models: one with 2/3/1 and one with
# 1/2/1 left/right/target populations
models = tibble(
           left = list(c('pop1', 'pop2'), c('pop3')),
           right = list(c('pop5', 'pop6', 'pop7'), c('pop7', 'pop8')),
           target = c('pop10', 'pop10'))
results = qpadm_multi('/my/geno/prefix', models)

## End(Not run)
```

---

qpadm_p *Faster version of* qpadm *with reduced output*

Description

Models target as a mixture of left populations, given a set of outgroup right populations. Can be used to estimate admixture proportions, and to estimate the number of independent admixture events.

Usage

```
qpadm_p(
  f2_data,
  left,
  right,
  target = NULL,
  auto_only = TRUE,
  fudge = 1e-04,
  boot = FALSE,
```

```
  constrained = FALSE,
  rnk = length(setdiff(left, target)) - 1,
  cpp = TRUE,
  weights = FALSE,
  f4blocks = NULL
)
```

Arguments

f2_data	Blocked f2-statistics (3d array), a directory path, or a genotype file prefix.
left	Left populations (sources)
right	Right populations (outgroups)
target	Target population
auto_only	Use only chromosomes 1 to 22.
fudge	Value added to diagonal matrix elements before inverting
boot	If FALSE (the default), block-jackknife resampling will be used to compute standard errors. Otherwise, block-bootstrap resampling will be used to compute standard errors. If boot is an integer, that number will specify the number of bootstrap resamplings. If boot = TRUE, the number of bootstrap resamplings will be equal to the number of SNP blocks.
constrained	Constrain admixture weights to be non-negative
rnk	Rank of f4-matrix. Defaults to one less than full rank.
cpp	Use C++ functions. Setting this to FALSE will be slower but can help with debugging.
weights	Return weights (default = FALSE)
f4blocks	Instead of f2 blocks, f4 blocks can be supplied. This is used by qpadm_multi

Value

Data frame with f4rank, dof, chisq, p, feasible

See Also

qpadm

Examples

```
left = c('Altai_Neanderthal.DG', 'Vindija.DG')
right = c('Chimp.REF', 'Mbuti.DG', 'Russia_Ust_Ishim.DG', 'Switzerland_Bichon.SG')
target = 'Denisova.DG'
qpadm_p(example_f2_blocks, left, right, target)
```

`qpadm_rotate` *Compute p-values for many qpadm models*

Description

This functions evaluates many qpadm models simultaneously by keeping the target population and the `rightfix` populations fixed, and distributing the `leftright` populations by keeping some in the set of left population and adding the remaining populations to the right populations. (See details for an example of how models are generated)

Usage

```
qpadm_rotate(
  f2_blocks,
  leftright,
  target,
  rightfix = NULL,
  full_results = FALSE,
  verbose = TRUE
)
```

Arguments

`f2_blocks`	3d array of blocked f2 statistics, output of `f2_from_precomp`.
`leftright`	Populations which will be distributed between left and right
`target`	Target population
`rightfix`	Populations which will be on the right side in all models
`full_results`	Return all output items which are returned by `qpadm`. By default (`full_results = FALSE`), weights and several other statistics will not be computed for each model, making it faster and the output more readable. If `full_results = TRUE`, the output will be a nested data frame where each row is one `qpadm` model, and each column has one data frame item from the regular qpadm output (`weights`, `f4`, `rankdrop`, `popdrop`).
`verbose`	Print progress updates

Details

If `leftright` consists of the populations L1, L2, L3, L4; `rightfix` is the population R; and `target` is T, the following models will be genrated:

(left), (right), (target)
(L1), (L2, L3, L4, R), (T)
(L2), (L1, L3, L4, R), (T)
(L3), (L1, L2, L4, R), (T)
(L4), (L1, L2, L3, R), (T)
(L1, L2), (L3, L4, R), (T)

(L1, L3), (L2, L4, R), (T)
(L1, L4), (L2, L3, R), (T)
(L2, L3), (L1, L4, R), (T)
(L2, L4), (L1, L3, R), (T)
(L3, L4), (L1, L2, R), (T)

Value

A data frame with Chi-squared statistics and p-values for each population combination

Examples

```
## Not run:
pops = dimnames(example_f2_blocks)[[1]]
qpadm_rotate(example_f2_blocks, leftright = pops[1:4],
             target = pops[5], rightfix = pops[6:7])

## End(Not run)
```

---

qpadm_sweep — *Sweep qpadm over a Cartesian product of targets, source-sets, and right-sets*

Description

Patterson-style sweeps fit qpadm for every combination of (target, source-set, right-set). Each invocation through `qpadm()` would re-load the f2 cache from disk; `qpadm_sweep()` loads it once via `qpadm_multi()` and returns a flat tibble with one row per combination, suitable for filtering / ranking model fits across a sweep without unnesting nested lists.

Usage

```
qpadm_sweep(
  data,
  targets,
  source_sets,
  right_sets,
  allsnps = FALSE,
  full_results = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`data`	The input data in the form of: • A 3d array of blocked f2 statistics, output of `f2_from_precomp` or `extract_f2` • A directory with f2 statistics • The prefix of a genotype file
`targets`	Character vector of target populations.
`source_sets`	Named list of character vectors; each element is a candidate set of source ("left") populations for one model. If unnamed, names default to `S1`, `S2`, ... Empty names are an error.
`right_sets`	Named list of character vectors; each element is a candidate set of "right" / outgroup populations. If unnamed, names default to `R1`, `R2`, ... Empty names are an error.
`allsnps`	Use all SNPs with allele frequency estimates in every population of any given population quadruple. If `FALSE` (the default) only SNPs which are present in all populations in `popcombs` (or any given model in it) will be used. When there are populations with lots of (non-randomly) missing data, `allsnps = TRUE` can lead to false positive results. This option only has an effect if `data` is the prefix of a genotype file. If `data` are f2-statistics, the behavior will be determined by the options that were used in computing the f2-statistics.
`full_results`	If `TRUE` (the default), the returned tibble includes list-columns `weights`, `rankdrop`, `popdrop`, and `f4_var_singular_loadings` with the full per-model output of `qpadm()`. If `FALSE`, only the flat summary columns are returned (including the always-on `f4_var_rcond` diagnostic).
`verbose`	Print progress updates
`...`	Further arguments forwarded to downstream functions. When `data` is a genotype-file prefix, kwargs matching `f4blockdat_from_geno()`'s formals are routed to the preflight f4 computation. Kwargs matching the per-fit function's formals (`qpadm` when `full_results=TRUE`, `qpadm_p()` when `FALSE`) are routed to each per-model fit. Unrecognised names raise a single error at entry naming both surfaces. `singular_threshold` (a `qpadm()` formal) is forwarded but be aware that if any model trips its threshold, `qpadm()` raises an error, the entire `qpadm_multi` call aborts via `furrr::future_map`, and no results are returned. For "mark and continue" semantics over a batch, omit `singular_threshold` and filter on `f4_var_rcond` after the call.

Details

This is a convenience wrapper around `qpadm_multi()` that adds:

- named source-sets and right-sets so each combination is labelled in the output
- implicit Cartesian product over `(targets x source_sets x right_sets)`
- a flat tibble result with top-level columns extracted from each model fit

For per-model parallel evaluation, set `future::plan('multisession')` before calling.

Value

A tibble with one row per `(target, source_set, right_set)` combination and columns:

- `target`, `source_set`, `right_set`: identifiers for the combination
- `left`, `right`: list-columns with the source / right pops for this model
- `f4rank`: tested rank in the top row of `qpadm()`'s `rankdrop` (= `length(left) - 1`)
- `p`, `chisq`, `dof`: top-row of `rankdrop` (the "auto" model fit)
- `feasible`: `TRUE` if all weights are between 0 and 1
- `f4_var_rcond`: scalar reciprocal-condition diagnostic of the f4 variance matrix used in the GLS solve. Values near machine epsilon (`~1e-15`) signal near-singular `f4_var` and warrant inspection of `f4_var_singular_loadings`. Always returned (not gated on `full_results`).
- `weights`: list-column with the per-source `weight` / `se` / `z` tibble (`full_results = TRUE`)
- `rankdrop`: list-column with the full rankdrop table (`full_results = TRUE`)
- `popdrop`: list-column with `qpadm()`'s leave-one-out per-source-pop fit table (`full_results = TRUE`). Always populated since `qpadm_sweep` always supplies a target.
- `f4_var_singular_loadings`: list-column (`full_results = TRUE`). Cells are tibbles identifying which right population is loading-heaviest on the singular direction; populated when the auto-bar fires (`f4_var_rcond < 1e-8`). Cells are `NULL` when the auto-bar does not fire (the common clean-data case) and also `NULL` if `qpadm()`'s eigendecomposition inside the loadings computation fails on a pathological matrix. Useful for loading-guided right-pop pruning.

Kwargs in `...` are forwarded to `qpadm_multi()`, which validates them upfront against the union of formals for `f4blockdat_from_geno()` (the geno-path preflight) and `qpadm()` (the per-fit call), then routes each to its respective downstream callee. Unrecognised names raise a single clear error at entry — see `qpadm_multi()`'s `...` docs for the full contract.

`singular_threshold` (a `qpadm()` formal, forwarded via `...`) does not produce a populated `f4_var_singular_loadings` cell in the sweep: `qpadm()` raises an error when the threshold trips, propagates through `furrr::future_map`, and aborts the entire sweep. For "mark and continue" semantics, omit `singular_threshold` and filter on `f4_var_rcond` after the sweep returns.

See Also

`qpadm()`, `qpadm_multi()`

Examples

```
## Not run:
# Run 3 targets * 2 source-sets * 2 right-sets = 12 qpadm models from one f2 dir.
targets    = c("Patterson_England_IA", "Patterson_England_BA",
               "Patterson_England_C_EBA")
sources    = list(canonical_3way = c("WHGA", "Balkan_N", "OldSteppe"),
                  with_ehg       = c("WHGA", "Balkan_N", "OldSteppe", "EHG_Karelia"))
rights     = list(distal_4pop    = c("OldAfrica", "WHGB", "Turkey_N", "Russia_Afanasievo"),
```

```
                    distal_refined = c("OldAfrica", "WHGB", "Turkey_N", "Russia_Afanasievo",
                                       "Iran_GanjDareh_N"))
res = qpadm_sweep("f2_dir/", targets, sources, rights)
res %>% arrange(target, p)

## End(Not run)
```

`qpadm_wrapper` *Wrapper function around the original qpAdm program*

Description

This requires a working installation of qpAdm, which will be called using `system`

Usage

```
qpadm_wrapper(
  pref,
  left,
  right,
  target = NULL,
  bin = "~np29/o2bin/qpAdm",
  outdir = "./",
  parfile = NULL,
  allsnps = "NO",
  blgsize = 0.05,
  fancyf4 = "NO",
  f4mode = "YES",
  inbreed = "NO",
  printonly = FALSE,
  env = "",
  verbose = TRUE
)
```

Arguments

`pref`	Path to and prefix of the packedancestrymap genotype files
`left`	Left populations (or leftlist file)
`right`	Right populations (or rightlist file)
`target`	Target population
`bin`	Path to the qpAdm binary file
`outdir`	Output directory. files `out`, `parfile`, `leftlist`, `rightlist` will be overwritten
`parfile`	qpAdm parameter file
`allsnps`	allsnps

`blgsize`	blgsize
`fancyf4`	fancyf4
`f4mode`	f4mode
`inbreed`	inbreed
`printonly`	Should the command be printed or executed?
`env`	Export environmental variables. See examples.
`verbose`	Print progress updates

Value

If not printonly, a data frame with parsed qpAdm output

Examples

```
## Not run:
left = c('Altai_Neanderthal.DG', 'Vindija.DG')
right = c('Chimp.REF', 'Mbuti.DG', 'Russia_Ust_Ishim.DG', 'Switzerland_Bichon.SG')
target = 'Denisova.DG'
qpadm_wrapper('genotype_prefix', left, right, target,
  bin = 'path/to/qpAdm')

## End(Not run)
```

---

qpdstat *Estimate f4 statistics*

Description

Computes f4-statistics of the form $f4(A, B; C, D)$. For allele frequencies `a`, `b`, `c`, `d`, `f4(A, B; C, D)` is computed as the average of `(a-b)*(c-d)` over all SNPs in each SNP block. This is equivalent to $(f2(A, D) + f2(B, C) - f2(A, C) - f2(B, D))/2$ (assuming no missing data). The input of this function can either be a 3d array of f2-statistics generated by `f2_from_precomp` or `f2_from_geno`, a directory with f2-statistics, or the prefix of genotype files. Computing f4 from genotype files directly is slower, but provides more flexibility in dealing with missing data (see details).

Usage

```
qpdstat(
  data,
  pop1 = NULL,
  pop2 = NULL,
  pop3 = NULL,
  pop4 = NULL,
  boot = FALSE,
  sure = FALSE,
```

```
  unique_only = TRUE,
  comb = TRUE,
  blgsize = NULL,
  block_lengths = NULL,
  f4mode = TRUE,
  afprod = TRUE,
  cpp = TRUE,
  verbose = is.character(data),
  ...
)
```

Arguments

`data`	Input data in one of three forms: 1. A 3d array of blocked f2 statistics, output of `f2_from_precomp` or `f2_from_geno` (fastest option) 2. A directory which contains pre-computed f2-statistics 3. The prefix of genotype files (slowest option)
`pop1`	One of the following four: 1. `NULL`: all possible population combinations will be returned 2. A vector of population labels. All combinations with the other `pop` arguments will be returned 3. A matrix with population combinations to be tested, with one population per column and one combination per row. Other `pop` arguments will be ignored. 4. the location of a file (`poplistname` or `popfilename`) which specifies the populations or population combinations to be tested. Other `pop` arguments will be ignored.
`pop2`	A vector of population labels
`pop3`	A vector of population labels
`pop4`	A vector of population labels
`boot`	If `FALSE` (the default), block-jackknife resampling will be used to compute standard errors. Otherwise, block-bootstrap resampling will be used to compute standard errors. If `boot` is an integer, that number will specify the number of bootstrap resamplings. If `boot = TRUE`, the number of bootstrap resamplings will be equal to the number of SNP blocks.
`sure`	The number of population combinations can get very large. This is a safety option that stops you from accidently computing all combinations if that number is large.
`unique_only`	If `TRUE` (the default), redundant combinations will be excluded
`comb`	Generate all combinations of `pop1`, `pop2`, `pop3`, `pop4`. If `FALSE`, `pop1`, `pop2`, `pop3`, `pop4` should all be vectors of the same length.
`blgsize`	SNP block size in Morgan. Default is 0.05 (5 cM). Only used when `data` is the prefix of genotype files

`block_lengths`	Vector with lengths of each jackknife block. `sum(block_lengths)` has to match the number of SNPs. only used when `data` is the prefix of genotype files
`f4mode`	Set this to `FALSE` to compute D-statistics instead of f4. This only has an effect if the first argument is a genotype prefix. D-statistics are computed as `(a-b)*(c-d) / ((a + b - 2*a*b) * (c + d - 2*c*d))`, which is the same as `(P(BABA) - P(ABBA)) / (P(ABBA) + P(BABA))`
`afprod`	Compute f4 from allele frequency products instead of f2 (default `TRUE`). Only used if `data` is a directory with precomputed data. For populations with lots of missing data, this option reduces bias that can result from setting `maxmiss` to values greater than 0. In all other cases it should not make a difference.
`cpp`	Use C++ functions. Setting this to `FALSE` will be slower but can help with debugging.
`verbose`	Print progress updates
`...`	Additional arguments passed to `f4blockdat_from_geno` if `data` is a genotype file prefix or `f2_from_precomp` if `data` is a directory with f2-statistics

Details

f4- and D-statistics are informative about how four populations are related to one another. Estimates of f4 are unbiased as long as assumptions about SNP ascertainment and mutation rates are met. Missing data can violate the ascertainment assumptions: an f4-statistic may be significantly different when it is calculated from all non-missing SNPs, compared to what it would be if it were calculated from all SNPs in the genome. However, because this difference is often small, f4 is often calculated using samples or populations with missing data, on a particular subset of all SNPs. There are different strategies for choosing the SNPs in this case, and these strategies differ in how many SNPs they use, how likely they lead to bias, and whether pre-computed f2-statistics can be used.

- Use the same SNPs for every f4-statistic
  This is the most conservative option, but also the option which will use the smallest number of SNPs, which may result in a lack of power. It is the default option when pre-computing f2-statistics (`maxmiss = 0` in `extract_f2` or `f2_from_geno`).
- Use different SNPs for each f4-statistic
  This option strikes a balance between avoiding bias and using a larger number of SNPs. For each f4-statistic it selects all SNPs which are present in all four populations. This option only works when the first argument is a genotype prefix (it doesn't work with pre-computed f2-statistics). In that case, this option is used by default. To turn it off and instead use the same SNPs for every f4-statistic, set `allsnps = FALSE`. This option is the default option in the original qpDstat program (it is in fact the only option for selecting SNPs in the original qpDstat; however in the original qpAdm and qpGraph programs, this mode of selecting SNPs can be enabled with the `allsnps` option, whereas the default mode in qpAdm and qpGraph is equivalent to `maxmiss = 0`)
- Use different SNPs for each f2-statistic
  This is the least conservative option, but also the option which uses most of the

available information. It makes it possible to use pre-computed f2-statistics for a large number of populations without losing a large number of SNPs. To use this option, set the `maxmiss` parameter to a value greater than 0 (and not larger than 1) in `extract_f2` or `f2_from_geno`. When using this option, be aware that bias is possible, in particular for f4-statistics where some populations have large amounts of missing data. To reduce the bias that can result from using this option, you may want to combine it with using the option `afprod = TRUE` in `f2_from_precomp`. In summary, whenever you work with populations with missing data, there is no guarantee that f4- or D-statistics involving these populations are not skewed in some way. If you choose to analyze these populations anyway, and you decide which SNPs to use, there is a trade-off between maximizing power and minimizing the risk of bias. One strategy might be to first use the least conservative option (setting `maxmiss = 1` in `extract_f2`) to get an overview, and then spot-check individual results using more conservative options.

Value

`qpdstat` returns a data frame with f4 statistics, with columns for populations 1 to 4, f4-estimate (`est`), standard error of the estimate (`se`), z-score (`z`, `est/se`), p-value (`2*(1-pnorm(z))`), and the number of SNPs used (`n`; only if first argument is genotype prefix)

Alias

`f4`

References

Patterson, N. et al. (2012) *Ancient admixture in human history* Genetics

Peter, B. (2016) *Admixture, Population Structure, and F-Statistics* Genetics

Examples

```
pop1 = 'Denisova.DG'
pop2 = c('Altai_Neanderthal.DG', 'Vindija.DG')
pop3 = c('Chimp.REF', 'Mbuti.DG', 'Russia_Ust_Ishim.DG')
pop4 = 'Switzerland_Bichon.SG'
qpdstat(example_f2_blocks, pop1, pop2, pop3, pop4)
## Not run:
qpdstat(f2_dir, pop1, pop2, pop3, pop4)

## End(Not run)
## Not run:
# compute D-statistics instead
qpdstat("/geno/prefix", pop1, pop2, pop3, pop4)

## End(Not run)
## Not run:
# make a data frame with the population combinations for which f4 should be computed
combinations = tibble(pop1 = pop1, pop2 = pop2[1], pop3 = pop3, pop4 = pop4)
qpdstat(example_f2_blocks, combinations)

## End(Not run)
```

`qpdstat_wrapper` *Wrapper function around the original qpDstat program*

Description

This requires a working installation of qpDstat, which will be called using `system`

Usage

```
qpdstat_wrapper(
  pref,
  pop1,
  pop2 = NULL,
  pop3 = NULL,
  pop4 = NULL,
  bin = "~np29/o2bin/qpDstat",
  outdir = ".",
  parfile = NULL,
  f4mode = "YES",
  inbreed = "NO",
  printonly = FALSE,
  env = "",
  verbose = TRUE
)
```

Arguments

`pref`	Path to and prefix of the packedancestrymap genotype files
`pop1`	One of the following four: 1. `NULL`: populations will be read from `poplistname` or `popfilename` specified in `parfile` 2. A vector of population labels 3. A data frame in which the first four columns specify the population combinations to be tested. `pop2`, `pop3`, `pop4` will be ignored. 4. the location of a file (`poplistname` or `popfilename`) which specifies the populations or population combinations to be tested. `pop2`, `pop3`, `pop4` will be ignored.
`pop2`	A vector of population labels
`pop3`	A vector of population labels
`pop4`	A vector of population labels
`bin`	Path to the qpDstat binary file
`outdir`	Output directory. files `out`, `parfile`, `poplistname`, `popfilename` may be overwritten
`parfile`	qpDstat parameter file. If this is specified, `pop`, `pop2`, `pop3`, and `pop4` will be ignored.

f4mode	f4mode
inbreed	inbreed
printonly	Should the command be printed or executed?
env	Export environmental variables. See examples.
verbose	Print progress updates

Value

If printonly, the qpDstat command, otherwise a data frame with parsed qpDstat output

Examples

```
## Not run:
pop1 = 'Denisova.DG'
pop2 = c('Altai_Neanderthal.DG', 'Vindija.DG')
pop3 = c('Chimp.REF', 'Mbuti.DG', 'Russia_Ust_Ishim.DG')
pop4 = 'Switzerland_Bichon.SG'
qpdstat_wrapper('genotype_prefix', pop1, pop2, pop3, pop4,
  bin = 'path/to/qpDstat', pref = 'path/to/packedancestrymap_prefix')

## End(Not run)
```

---

qpf4diff *Estimate f4 differences*

Description

Estimate f4 differences

Usage

```
qpf4diff(data, pops, boot = FALSE, verbose = FALSE)
```

Arguments

data	Input data in one of three forms: 1. A 3d array of blocked f2 statistics, output of f2_from_precomp or f2_from_geno (fastest option) 2. A directory which contains pre-computed f2-statistics 3. The prefix of genotype files (slowest option)
pops	A vector of 5 populations or a five column population matrix. The following ratios will be computed: f4(1, 2; 3, 4)/f4(1, 2; 5, 4)
boot	If FALSE (the default), block-jackknife resampling will be used to compute standard errors. Otherwise, block-bootstrap resampling will be used to compute standard errors. If boot is an integer, that number will specify the number of bootstrap resamplings. If boot = TRUE, the number of bootstrap resamplings will be equal to the number of SNP blocks.
verbose	Print progress updates

Value

`qpf4diff` returns a data frame with f4 ratios

`qpf4ratio` *Estimate admixture proportions via f4 ratios*

Description

Estimate admixture proportions via f4 ratios

Usage

```
qpf4ratio(
  data,
  pops,
  auto_only = TRUE,
  blgsize = 0.05,
  poly_only = FALSE,
  boot = FALSE,
  verbose = FALSE
)
```

Arguments

`data`	Input data in one of three forms: 1. A 3d array of blocked f2 statistics, output of `f2_from_precomp` or `f2_from_geno` (fastest option) 2. A directory which contains pre-computed f2-statistics 3. The prefix of genotype files (slowest option)
`pops`	A vector of 5 populations or a five column population matrix. The following ratios will be computed: `f4(1, 2; 3, 4)/f4(1, 2; 5, 4)`
`auto_only`	Use only chromosomes 1 to 22.
`blgsize`	SNP block size in Morgan. Default is 0.05 (5 cM). If `blgsize` is 100 or greater, if will be interpreted as base pair distance rather than centimorgan distance.
`poly_only`	Exclude sites with identical allele frequencies in all populations.
`boot`	If `FALSE` (the default), block-jackknife resampling will be used to compute standard errors. Otherwise, block-bootstrap resampling will be used to compute standard errors. If `boot` is an integer, that number will specify the number of bootstrap resamplings. If `boot = TRUE`, the number of bootstrap resamplings will be equal to the number of SNP blocks.
`verbose`	Print progress updates

Value

`qpf4ratio` returns a data frame with f4 ratios

`qpf4ratio_wrapper` *Wrapper function around the original qpF4ratio program*

Description

This requires a working installation of qpF4ratio, which will be called using `system`

Usage

```
qpf4ratio_wrapper(
  pref,
  pops,
  bin = "~np29/o2bin/qpF4ratio",
  outdir = ".",
  parfile = NULL,
  blgsize = 0.05,
  fancyf4 = "YES",
  printonly = FALSE,
  env = "",
  verbose = TRUE
)
```

Arguments

`pref`	Path to and prefix of the packedancestrymap genotype files
`pops`	A vector of five populations, or a 5 x n matrix with population names. For each line `alpha` will be computed as `f4(1,2; 3,4)/f4(1,2; 5,4)`
`bin`	Path to the qpF4ratio binary file
`outdir`	Output directory. files `out`, `parfile`, `poplistname`, `popfilename` may be overwritten
`parfile`	qpF4ratio parameter file. If this is specified, `pops` will be ignored.
`blgsize`	blgsize
`fancyf4`	fancyf4
`printonly`	Should the command be printed or executed?
`env`	Export environmental variables. See examples.
`verbose`	Print progress updates

Value

If `printonly`, the `qpF4ratio` command, otherwise a data frame with parsed `qpF4ratio` output

Examples

```
## Not run:
pops = c('Denisova.DG', 'Altai_Neanderthal.DG', 'Vindija.DG', 'Chimp.REF', 'Mbuti.DG')
qpf4ratio_wrapper('genotype_prefix', pops, bin = 'path/to/qpDstat')

## End(Not run)
```

---

qpfstats *Get smoothed f2-statistics*

Description

This function returns an array of (pseudo-) f2-statistics which are computed by taking into account other f2-, f3-, and f4-statistics. The advantage of doing that is that f3- and f4-statistics computed from these smoothed f2-statistics can be more accurate for populations with large amounts of missing data. The function uses SNPs which are missing in some populations in a manner which tends to introduce less bias than setting `maxmiss` to values greater than 0.

Usage

```
qpfstats(
  pref,
  pops,
  include_f2 = TRUE,
  include_f3 = TRUE,
  include_f4 = TRUE,
  verbose = TRUE
)
```

Arguments

`pref`	Prefix of genotype files
`pops`	Populations for which to compute f2-statistics
`include_f2`	Should f2-statistics be used to get smoothed f2-statistics? If `include_f2` is a positive integer, it specifies how many randomly chosen f2-statistics should be used.
`include_f3`	Should f3-statistics be used to get smoothed f2-statistics? If `include_f3` is a positive integer, it specifies how many randomly chosen f3-statistics should be used.
`include_f4`	Should f4-statistics be used to get smoothed f2-statistics? If `include_f4` is a positive integer, it specifies how many randomly chosen f4-statistics should be used.
`verbose`	Print progress updates (default `TRUE`).

Value

A 3d-array of smoothed f2-statistics

Examples

```
## Not run:
f2_blocks = qpfstats(geno_prefix, mypops)

## End(Not run)
```

---

qpgraph *Compute the fit of an admixture graph*

Description

Computes the fit of a given admixturegraph from f2-statistics. Drift edge weights and admixture edges weights are optimized until the (negative) likelihood score is minimized. The likelihood score is based on the squared difference between estimated and fitted f3-statistics.

Usage

```
qpgraph(
  data,
  graph,
  lambdascale = 1,
  boot = FALSE,
  diag = 1e-04,
  diag_f3 = 1e-05,
  lsqmode = FALSE,
  numstart = 10,
  seed = NULL,
  cpp = TRUE,
  return_fstats = FALSE,
  return_pvalue = FALSE,
  f3precomp = NULL,
  f3basepop = NULL,
  constrained = TRUE,
  allsnps = FALSE,
  ppinv = NULL,
  f2_blocks_test = NULL,
  verbose = FALSE
)
```

Arguments

`data`	Input data in one of three forms: 1. A 3d array of blocked f2 statistics, output of `f2_from_precomp` or `extract_f2` (fastest option) 2. A directory which contains pre-computed f2-statistics 3. The prefix of genotype files (slowest option)
`graph`	An admixture graph represented as a matrix of edges, an `igraph` object, or the path to a *qpGraph* graph file. Edges can be constrained by providing a matrix or data frame of edges with columns titled `lower` and `upper` with lower and upper bounds, respectively. By default, admixture edges are constrained to be between zero and one (with paired edges summing to one), and drift edges have a lower bound at zero.
`lambdascale`	Scales f2-statistics. This has no effect on the fit, but is used in the original *qpGraph* program to display branch weights on a scale that corresponds to FST distances.
`boot`	If `FALSE` (the default), each block will be left out at a time and the covariance matrix of f3 statistics will be computed using block-jackknife. Otherwise bootstrap resampling is performed `n` times, where `n` is either equal to `boot` if it is an integer, or equal to the number of blocks if `boot` is `TRUE`. The covariance matrix of f3 statistics will be computed using bootstrap resampling.
`diag`	Regularization term added to the diagonal elements of the covariance matrix of fitted branch lengths (after scaling by the matrix trace). Default is 0.0001.
`diag_f3`	Regularization term added to the diagonal elements of the covariance matrix of estimated f3 statistics (after scaling by the matrix trace). In the original *qpGraph* program, this is fixed at 0.00001.
`lsqmode`	Least-squares mode. If `TRUE`, the likelihood score will be computed using a diagonal matrix with `1/(sum(diag(f3_var)) * diag_f3)`, in place of the inverse f3-statistic covariance matrix. `lsqmode = 2` will use the identity matrix instead, which is equivalent to computing the score as the sum of squared residuals (`sum((f3_est-f3_fit)^2)`). Both of these options do not take the covariance of f3-statistics into account. This can lead to bias, but is more stable in cases where the inverse f3-statistics covariance matrix can not be estimated precisely (for example because the number of populations is large). An alternative to `lsqmode = TRUE` that doesn't completely ignore the covariance of f3-statistics is to increase `diag_f3`.
`numstart`	Number of random initializations of starting weights. Defaults to 10. Increasing this number will make the optimization slower, but reduce the risk of not finding the optimal weights. Check the `opt` output to see how much the optimization depends on the starting weights.
`seed`	Random seed for generating starting weights.
`cpp`	Use C++ functions. Setting this to `FALSE` will be slower but can help with debugging.

`return_fstats` Return estimated and fitted f2- and f4-statistics, as well as the worst f4-statistic residual Z-score. Defaults to `FALSE` because this can be slow.

`return_pvalue` Return a p-value for the graph fit based on the jackknife score distribution (default `FALSE`).

`f3precomp` Optional precomputed f3-statistics. This should be the output of `qpgraph_precompute_f3` and can be provided instead of `data`. This can speed things up if many graphs are evaluated using the same set of f3-statistics.

`f3basepop` Optional f3-statistics base population. Inference will be based on f3-statistics of the form `f3(f3basepop; i, j)` for all population pairs `(i, j)`. Defaults to the outgroup population if the graph has one. This option is ignored if `f3precomp` is provided. Changing `f3basepop` should make very little difference.

`constrained` Constrain estimated drift edge weights to be non-negative, and admixture edge weights to be between zero and one.

`allsnps` Compute f3 from different SNPs for each population triplet (if data is missing for some SNPs and populations). This only has an effect when `data` is the prefix of genotype files.

`ppinv` Optional inverse f3-statistics covariance matrix

`f2_blocks_test`

An optional 3d array of f2-statistics used for computing an out-of-sample score. This should contain only SNP blocks which are not part of `f2_blocks`. This allows to estimate the fit of a graph without overfitting and will not be used during the optimization step

`verbose` Print progress updates

Value

`qpgraph` returns a list with data describing the model fit:

- `edges`: A data frame where each row is an edge in the graph. For regular edges, the column `weight` is the estimated edge length, and for admixture edges, it is the estimated admixture weight.
- `score`: The likelihood score of the fitted graph. Lower values correspond to better fits. The score is calculated as the inner product of the residuals (difference between estimated and fitted f3 statistics), weighted by the inverse of the f3 covariance matrix. See `qpgraph_score`
- `f2`: Estimated and fitted f2 statistics (if `return_fstats = TRUE`). p-values and z-scores test the significance of the difference.
- `f3`: Estimated and fitted f3 statistics. p-values and z-scores test the significance of the difference.
- `f4`: Estimated and fitted f4 statistics (if `return_fstats = TRUE`). p-values and z-scores test the significance of the difference.
- `opt`: A data frame with details of the weight-fitting step, including the randomly sampled starting weights. The column `value` contains the score for each set of starting weights. Columns starting with `x` denote initial weights, and columns starting with `y` denote fitted weights.

- `worst_residual`: The highest absolute z-score of f4-statistics residuals (fitted - estimated f4); (returned if `return_fstats = TRUE`)

References

Patterson, N. et al. (2012) *Ancient admixture in human history.* Genetics

See Also

`qpgraph_wrapper` for a wrapper functions which calls the original *qpGraph* program.

Examples

```
out = qpgraph(example_f2_blocks, example_graph)
plot_graph(out$edges)
```

---

qpgraph_precompute_f3

Compute f3-statistics from f2-statistics.

Description

Takes a 3d array of f2 block jackknife estimates and computes f3-statistics between the first population $p1$ and all population pairs i, j: $f3(p1; p_i, p_j)$

Usage

```
qpgraph_precompute_f3(
  data,
  pops,
  f3basepop = NULL,
  lambdascale = 1,
  boot = FALSE,
  seed = NULL,
  diag_f3 = 1e-05,
  lsqmode = FALSE
)
```

Arguments

`data`	Input data in one of three forms: 1. A 3d array of blocked f2 statistics, output of `f2_from_precomp` or `extract_f2` (fastest option) 2. A directory which contains pre-computed f2-statistics 3. The prefix of genotype files (slowest option)
`pops`	Populations for which to compute f3-statistics

`f3basepop` f3-statistics base population. If `NULL` (the default), the first population in `pops` will be used as the basis.

`lambdascale` Scales f2-statistics. This has no effect on the fit, but is used in the original *qpGraph* program to display branch weights on a scale that corresponds to FST distances.

`boot` If `FALSE` (the default), block-jackknife resampling will be used to compute standard errors. Otherwise, block-bootstrap resampling will be used to compute standard errors. If `boot` is an integer, that number will specify the number of bootstrap resamplings. If `boot = TRUE`, the number of bootstrap resamplings will be equal to the number of SNP blocks.

`seed` Random seed used if `boot` is `TRUE`.

`diag_f3` Regularization term added to the diagonal elements of the covariance matrix of estimated f3 statistics (after scaling by the matrix trace). In the original *qpGraph* program, this is fixed at 0.00001.

`lsqmode` Least-squares mode. If `TRUE`, the likelihood score will be computed using a diagonal matrix with `1/(sum(diag(f3_var)) * diag_f3)`, in place of the inverse f3-statistic covariance matrix. `lsqmode = 2` will use the identity matrix instead, which is equivalent to computing the score as the sum of squared residuals (`sum((f3_est-f3_fit)^2)`). Both of these options do not take the covariance of f3-statistics into account. This can lead to bias, but is more stable in cases where the inverse f3-statistics covariance matrix can not be estimated precisely (for example because the number of populations is large). An alternative to using `lsqmode = TRUE` which doesn't completely ignore the covariance of f3-statistics is to increase `diag_f3`.

Value

A list with four items

1. `f3_est` a matrix with f3-statistics for all population pairs with the output
2. `ppinv` a matrix with the inverse of the f3-statistic covariance matrix
3. `f2out` a data frame with f2 estimates
4. `f3out` a data frame with f3 estimates

Examples

```
pops = get_leafnames(example_igraph)
qpgraph_precompute_f3(example_f2_blocks, pops)$f3_est
## Not run:
qpgraph_precompute_f3(f2_dir, pops)

## End(Not run)
```

qpgraph_resample_multi *Evaluate a qpgraph models many times*

Description

This function is used in combination with compare_fits in order to test whether one graph has a significantly better fit than another. It creates bootstrap resampled SNP block training and test sets, and uses them to evaluate multiple graphs.

Usage

```
qpgraph_resample_multi(f2_blocks, graphlist, nboot, verbose = TRUE, ...)
```

Arguments

f2_blocks	3d array of f2-statistics
graphlist	A list of admixture graphs
nboot	Number of bootstrap iterations
verbose	Print progress updates
...	Arguments passed to qpgraph

Value

A list of same length as graphlist with data frames with qpgraph results for each iteration of bootstrap resampled f2-statistics

See Also

compare_fits

Examples

```
## Not run:
fits = qpgraph_resample_multi(f2_blocks, list(graph1, graph2), nboot = 100)
compare_fits(fits[[1]]$score_test, fits[[2]]$score_test)

## End(Not run)
```

qpgraph_resample_snps2 *Evaluate a qpgraph model many times*

Description

This function is used in combination with `compare_fits` in order to test whether one graph has a significantly better fit than another. using a list of bootstrap resampled f2-statistics and corresponding test-set f2-statistics

Usage

```
qpgraph_resample_snps2(f2_blocks, graph, f2_blocks_test, verbose = TRUE, ...)
```

Arguments

`f2_blocks`	A list of bootstrap resampled f2-statistics
`graph`	An admixture graph
`f2_blocks_test`	A list of f2-statistics from all blocks which were not used in the corresponding f2-array in `f2_blocks`
`verbose`	Print progress updates
`...`	Arguments passed to `qpgraph`

Value

A data frame with `qpgraph` results for each iteration of bootstrap resampled f2-statistics

See Also

`compare_fits` `boo_list`

qpgraph_wrapper *Wrapper function around the original qpGraph program*

Description

Wrapper function around the original qpGraph program

Usage

```
qpgraph_wrapper(
  pref,
  graph,
  bin = "~np29/o2bin/qpGraph",
  parfile = NULL,
  outdir = ".",
  printonly = FALSE,
  badsnps = NULL,
  lambdascale = -1,
  inbreed = "NO",
  diag = 1e-04,
  blgsize = 0.05,
  outpop = "NULL",
  loadf3 = NULL,
  lsqmode = "NO",
  fstdmode = "NO",
  hires = "NO",
  forcezmode = "NO",
  zthresh = 0,
  allsnps = "NO",
  oldallsnps = "NO",
  doanalysis = "YES",
  bigiter = 100,
  initmix = 0,
  env = "",
  verbose = TRUE
)
```

Arguments

`pref`	Prefix of the packedancestrymap format genotype files.
`graph`	An admixture graph or qpGraph graph file
`bin`	Location of the qpGraph binary
`parfile`	qpGraph parameter file
`outdir`	Output directory
`printonly`	Should output be executed or the command just be printed?
`badsnps`	badsnps
`lambdascale`	lambdascale
`inbreed`	inbreed
`diag`	diag
`blgsize`	blgsize
`outpop`	outgroup population
`loadf3`	loadf3

`lsqmode`	least-squares mode. sets the offdiagonal elements of the block-jackknife covariance matrix to zero.
`fstdmode`	fstdmode
`hires`	hires
`forcezmode`	forcezmode
`zthresh`	zthresh
`allsnps`	allsnps
`oldallsnps`	oldallsnps
`doanalysis`	doanalysis
`bigiter`	bigiter
`initmix`	initmix
`env`	Export environmental variables. See examples.
`verbose`	Print progress updates

Value

A list with parsed qpGraph output

1. `edges`: data frame
2. `score`: scalar
3. `f2`: data frame

Examples

```
## Not run:
qpgraph_wrapper('genotype_prefix', example_graph,
                bin = 'path/to/qpGraph')

## End(Not run)
```

`qpwave` *Estimate admixture waves*

Description

`qpwave` compares two sets of populations (`left` and `right`) to each other. It estimates a lower bound on the number of admixtue waves that went from `left` into `right`, by comparing a matrix of f4-statistics to low-rank approximations. For a rank of 0 this is equivalent to testing whether `left` and `right` form clades relative to each other.

Usage

```
qpwave(
  data,
  left,
  right,
  fudge = 1e-04,
  auto_only = TRUE,
  blgsize = 0.05,
  poly_only = FALSE,
  boot = FALSE,
  constrained = FALSE,
  cpp = TRUE,
  singular_threshold = NA_real_,
  verbose = TRUE
)
```

Arguments

`data`	The input data in the form of: • A 3d array of blocked f2 statistics, output of `f2_from_precomp` or `extract_f2` • A directory with f2 statistics • The prefix of a genotype file
`left`	Left populations (sources)
`right`	Right populations (outgroups)
`fudge`	Value added to diagonal matrix elements before inverting
`auto_only`	Use only chromosomes 1 to 22.
`blgsize`	SNP block size in Morgan. Default is 0.05 (5 cM). If `blgsize` is 100 or greater, if will be interpreted as base pair distance rather than centimorgan distance.
`poly_only`	Exclude sites with identical allele frequencies in all populations.
`boot`	If `FALSE` (the default), block-jackknife resampling will be used to compute standard errors. Otherwise, block-bootstrap resampling will be used to compute standard errors. If `boot` is an integer, that number will specify the number of bootstrap resamplings. If `boot = TRUE`, the number of bootstrap resamplings will be equal to the number of SNP blocks.
`constrained`	Constrain admixture weights to be non-negative
`cpp`	Use C++ functions. Setting this to `FALSE` will be slower but can help with debugging.
`singular_threshold`	If not `NA` (the default), error out when the reciprocal condition number of the (fudged) f4 variance matrix is below this threshold. A common choice is `1e-12` – below that the downstream `solve()` and the C++ weight optimization fall back to a pseudo-inverse, which makes the returned weight standard errors numerically tight in a way that doesn't reflect actual

sampling uncertainty. The default (`NA`) preserves the historical behavior: silently fall through to the pseudo-inverse fallback. In either mode, the reciprocal condition number is reported as `f4_var_rcond` on the returned list so callers can gate downstream interpretations programmatically.

Low-level RcppArmadillo `solve(): system is singular; attempting approx solution` stderr lines are suppressed at compile time via `ARMA_WARN_LEVEL 1`. Those lines previously flooded stderr on batch fits at higher K without carrying any information not already exposed by the R-side diagnostics. The canonical programmatic signal is `f4_var_rcond` (always populated, regardless of `verbose`); `qpadm_multi()` and `qpadm_sweep()` force `verbose = FALSE` internally, so the interactive `warning()` only fires for direct `qpadm()` calls at `verbose = TRUE`. Batch callers should inspect `f4_var_rcond` (and `f4_var_singular_loadings`) on the returned object rather than rely on stderr / warning output.

`verbose` Print progress updates

Value

`qpwave` returns a list with up to two data frames describing the model fit:

1. `f4` A data frame with estimated f4-statistics
2. `rankdrop`: A data frame describing model fits with different ranks, including *p*-values for the overall fit and for nested models (comparing two models with rank difference of one). A model with `L` left populations and `R` right populations has an f4-matrix of dimensions `(L-1)*(R-1)`. If no two left population form a clade with respect to all right populations, this model will have rank `(L-1)*(R-1)`.
   - `f4rank`: Tested rank
   - `dof`: Degrees of freedom of the chi-squared null distribution: `(L-1-f4rank)*(R-1-f4rank)`
   - `chisq`: Chi-sqaured statistic, obtained as `E'QE`, where `E` is the difference between estimated and fitted f4-statistics, and `Q` is the f4-statistic covariance matrix.
   - `p`: p-value obtained from `chisq` as `pchisq(chisq, df = dof, lower.tail = FALSE)`
   - `dofdiff`: Difference in degrees of freedom between this model and the model with one less rank
   - `chisqdiff`: Difference in chi-squared statistics
   - `p_nested`: *p*-value testing whether the difference between two models of rank difference 1 is significant

References

Patterson, N. et al. (2012) *Ancient admixture in human history.* Genetics

Haak, W. et al. (2015) *Massive migration from the steppe was a source for Indo-European languages in Europe.* Nature (SI 10)

See Also

qpadm

Examples

```
left = c('Altai_Neanderthal.DG', 'Vindija.DG')
right = c('Chimp.REF', 'Mbuti.DG', 'Russia_Ust_Ishim.DG', 'Switzerland_Bichon.SG')
qpwave(example_f2_blocks, left, right)
```

---

qpwave_pairs *Compute all pairwise qpwave p-values*

Description

For all pairs of left populations qpwave rank 0 Chi-squared statistics and p-values will be computed. This tests for each pair of left populations whether it forms a clade with respect to the right populations.

Usage

```
qpwave_pairs(f2_data, left, right)
```

Arguments

f2_data	A 3d array of blocked f2 statistics, output of f2_from_precomp. Alternatively, a directory with f2 statistics.
left	Left populations
right	Right populations

Examples

```
## Not run:
left = pops[5:7]
right = pops[1:4]
f2_blocks = f2_from_precomp('/my/f2/dir/', pops = left, pops2 = right, afprod = TRUE)
qpwave_pairs(f2_blocks, left, right)
# If f2-stats are too big to load them into memory,
# the following will read the data for each pair from disk:
qpwave_pairs('/my/f2/dir/', left, right)

## End(Not run)
```

`random_admixturegraph` *Generate a random admixture graph*

Description

This function randomly generates an admixture graph for a given set of leaf nodes

Usage

```
random_admixturegraph(
  leaves,
  numadmix = 0,
  simple = TRUE,
  outpop = NULL,
  nonzero_f4 = NULL,
  admix_constraints = NULL,
  event_order = NULL,
  ntry = 100
)
```

Arguments

`leaves`	Names of the leaf nodes, or a number specifying how many leaf nodes there should be
`numadmix`	Number of admixture events
`simple`	Should edges leading to admixture nodes consist of separate admix edges and normal edges
`outpop`	Outgroup population
`nonzero_f4`	Optional constraint: population pairs for which f4 must be non-zero.
`admix_constraints`	Optional constraint on admixture counts per population (see `satisfies_constraints`).
`event_order`	Optional constraint on the ordering of demographic events (see `satisfies_constraints`).
`ntry`	Maximum number of attempts to generate a graph satisfying all constraints (default 100).

Examples

```
rand_graph = random_admixturegraph(10, numadmix = 5)
plot_graph(rand_graph)
```

random_dates — *Get random dates for graph nodes*

Description

This function assigns a date (in generations) to each node in an admixture graph and is used in `random_sim`. The dates are drawn from a uniform distribution with given lower and upper bounds, unless the `fix_leaf` option is set to `TRUE`, in which case the dates of all leaf nodes returned by `get_leafnames` will be set to 0.

Usage

```
random_dates(graph, min = 1000, max = 1000, fix_leaf = FALSE)
```

Arguments

`graph`	An admixture graph
`min`	A lower limit of uniform distribution
`max`	An upper limit of uniform distribution
`fix_leaf`	A boolean specifying if the dates of the leaf nodes will be fixed at time 0 (i.e., at the most recent time). If `TRUE`, all samples will be drawn at the end of the simulation (i.e., from "today"). The default is `FALSE`.

Value

A named vector with random dates for each graph node

random_newick — *Generate a random binary graph*

Description

Generate a random binary graph

Usage

```
random_newick(n, start = "", end = "", outpop = NULL)
```

Arguments

`n`	The number of terminal nodes, or a vector of population labels.
`start`	Prefix.
`end`	Postfix.
`outpop`	Outgroup (if `n` is a vector of labels).

Value

Tree in newick format.

Examples

```
random_newick(5)
random_newick(c('a', 'b', 'c', 'd')) # toplogy random, but pop order fixed
random_newick(sample(c('a', 'b', 'c', 'd'))) # toplogy and pop order random
```

---

random_sim *Generate a random graph and simulate it in msprime v1.x*

Description

This is basically a wrapper function around the msprime_genome that allows user to create a random graph and simulate it in msprime v1.x.

Usage

```
random_sim(
  nleaf,
  nadmix,
  outpref = "random_sim",
  max_depth = NULL,
  ind_per_pop = 1,
  mutation_rate = 1.25e-08,
  admix_weights = 0.5,
  neff = 1000,
  time = 1000,
  fix_leaf = FALSE,
  outpop = NULL,
  nchr = 1,
  recomb_rate = 2e-08,
  seq_length = 1000,
  ghost_lineages = TRUE,
  run = FALSE
)
```

Arguments

nleaf	The number of leaf nodes
nadmix	The number of admixture events
outpref	A prefix of output files
max_depth	A constraint specifying the maximum time depth of the admixture graph (in generations)

`ind_per_pop` The number of individuals to simulate for each population. If a scalar value, it will be constant across all populations. Alternatively, it can be a named vector with a different value for each population.

`mutation_rate` Mutation rate per site per generation. The default is `1.25e-8` per base pair per generation.

`admix_weights` Admixture weights. If a float value ($0 <$ value < 1), admixture weights for each admixture event will be (value, 1-value). Alternatively, it can be a range, i.e., `c(0.1, 0.4)` specifying lower and upper limits of a uniform distribution from which the admixture weight value will be drawn. By default, all admixture edges have a weight of 0.5.

`neff` Effective population size (in diploid individuals). If a scalar value, it will be constant across all populations. Alternatively, it can be a range, i.e., `c(500, 1000)` specifying lower and upper limits of an uniform distribution from which values will be drawn

`time` Time between nodes. Either a scalar value (1000 by default) with the dates generated by `pseudo_dates`, or a range, i.e., `c(500, 1000)` specifying lower and upper limits of a uniform distribution from which values will be drawn (see `random_dates`)

`fix_leaf` A boolean specifying if the dates of the leaf nodes will be fixed at time 0. If `TRUE`, all samples will be drawn at the end of the simulation (i.e., from "today").

`outpop` A name of the (optional) outgroup population.

`nchr` The number of chromosomes to simulate

`recomb_rate` A float value specifying recombination rate along the chromosomes. The default is `2e-8` per base pair per generation.

`seq_length` The sequence length of the chromosomes. If it is a scalar value, the sequence length will be constant for all chromosomes. Alternatively, it can be a vector with a length equal to number of chromosomes (i.e., `c(100,50)` to simulate 2 chromosomes with the lengths of 100 and 50 base pairs).

`ghost_lineages`
A boolean value specifying whether ghost lineages will be allowed. If `TRUE`, admixture happens at the time points defined by the y-axis generated while plotting the graph by `plot_graph` If `FALSE` (default), admixture occurs at the time of the previous split event

`run` If `FALSE`, the function will terminate after writing the msprime script. If `TRUE`, it will try and execute the function with the default python installation. If you want to use some other python installation, you can set `run = /my/python`.

Value

A list with the path of simulation script, a data frame of graph edges, dates and population sizes:

out A file name and path of the simulation script

edges An edge dataframe with admixture weights

dates A named vector with a date for each node

neffs A named vector with an effective population size for each node

Examples

```
# Create simulation script that simulates 2 chromosomes that are 50base long
# where maximum depth of the tree is 5000 generations, and plot the output graph
## Not run:
out = random_sim(nleaf=4, nadmix=0, max_depth=5000, nchr=2, seq_length=50)
plot_graph(out$edges, dates = out$dates, neff = out$neff, hide_weights = TRUE)

## End(Not run)
```

---

read_eigenstrat *Read genotype data from* EIGENSTRAT *files*

Description

Read genotype data from *EIGENSTRAT* files

Usage

```
read_eigenstrat(
  pref,
  inds = NULL,
  pops = NULL,
  first = 1,
  last = Inf,
  transpose = FALSE,
  verbose = TRUE
)
```

Arguments

`pref`	Prefix of the packedancestrymap files
`inds`	Individuals for which data should be read. Defaults to all individuals
`pops`	Populations for which data should be read. Cannot be provided together with 'inds'
`first`	Index of first SNP to read
`last`	Index of last SNP to read
`transpose`	Transpose genotype matrix (default is `snps` x `individuals`)
`verbose`	Print progress updates

Value

A list with the genotype data matrix, the `.ind` file, and the `.snp` file

Examples

```
## Not run:
samples = c('Ind1', 'Ind2', 'Ind3')
geno = read_packedancestrymap(prefix, samples)

## End(Not run)
```

`read_f2` *Read blocked f2 estimates from disk*

Description

This function reads blocked f2 estimates (or allele frequency products) which were writtend to disk by `write_f2` and returns them as a 3d array.

Usage

```
read_f2(
  f2_dir,
  pops = NULL,
  pops2 = NULL,
  type = "f2",
  counts = FALSE,
  remove_na = TRUE,
  verbose = FALSE
)
```

Arguments

`f2_dir`	Directory from which to read files
`pops`	Populations for which f2 statistics should be read. Defaults to all populations, which may require a lot of memory.
`pops2`	Specify this if you only want to read a subset of all population pairs. The resulting array will differ on 1st and 2nd dimension and will not work with all functions.
`type`	Type of statistic to read: `'f2'` (default), `'ap'` (allele frequency products), or `'fst'`.
`counts`	Return allele counts instead of f2 estimates
`remove_na`	Remove blocks with missing values
`verbose`	Print progress updates

Value

A 3d array of block jackknife estimates

See Also

write_f2

Examples

```
## Not run:
read_f2(f2_dir, pops = c('pop1', 'pop2', 'pop3'))

## End(Not run)
```

`read_f2_cache_metadata` *Read the cache metadata sidecar written by extract_f2()*

Description

Reads the `cache_metadata.json` file written by `extract_f2()` into the f2 output directory and returns it as a named R list. This gives wrappers and orchestrators a stable API for post-build telemetry (`n_snps`, `n_blocks`, `pops`, ...) without reaching into the internal `*.rds` layout.

Usage

```
read_f2_cache_metadata(outdir)
```

Arguments

`outdir`	Path to the f2 output directory passed to `extract_f2()`.

Details

The exact set of fields depends on which code path produced the cache:

- **Regular path** (`qpfstats = FALSE`): `schema_version`, `admixtools_version`, `built_at`, `pops`, `n_snps`, `n_blocks`, `blgsize`, `maxmiss`, `minmaf`, `maxmaf`, `minac2`, `auto_only`, `transitions`, `transversions`, `adjust_pseudohaploid`, `poly_only`, `apply_corr`, `afprod`, `fst`, `qpfstats` (= `FALSE`), `cache_id`.
- **qpfstats path** (`qpfstats = TRUE`): `schema_version`, `admixtools_version`, `built_at`, `pops`, `n_snps`, `n_blocks`, `blgsize`, `maxmiss`, `adjust_pseudohaploid`, `qpfstats` (= `TRUE`), `cache_id`. Filter arguments that qpfstats handles internally (`minmaf`, `maxmaf`, `auto_only`, `transitions`, etc.) are omitted because they don't apply on this code path.

`n_snps` is the count of SNPs that actually contributed to the f2 blocks (i.e. `sum(block_lengths_f2)`), not the post-filter row count of the snpfile. With the default `poly_only = c('f2')`, non-polymorphic SNPs survive filtering but are excluded from f2 blocks, so `n_snps` can be smaller than the visible row count in `snpdat.tsv.gz`. Both the regular and qpfstats paths report `n_snps` with this same definition.

`cache_id` is `NULL` (rendered as JSON `null`) if the cache-id computation failed at build time; `built_at` is ISO 8601 with millisecond precision in UTC. The `cache_id` value mirrors the `.f2_cache_id` sidecar contents.

`cache_metadata.json` is written via tempfile + `file.rename`, which is atomic on POSIX filesystems: a SIGKILL mid-`extract_f2` either leaves the previous version intact or commits the new one — never a truncated file.

Value

A named list with the fields described above.

See Also

`extract_f2()`, `compute_f2_cache_id()`, `result_to_json()`

Examples

```
## Not run:
extract_f2("my_geno_prefix", "f2_out/")
meta = read_f2_cache_metadata("f2_out/")
meta$n_snps      # SNPs that contributed to f2 blocks
meta$n_blocks    # number of jackknife blocks
meta$cache_id    # matches the .f2_cache_id sidecar

## End(Not run)
```

`read_lgo` *Parse a LEGOFIT .lgo file (minimum-viable subset)*

Description

Parses the subset of LEGOFIT 1.87 `.lgo` grammar that `graph_to_lgo()` emits. Full grammar coverage (line continuation, `constrained` expressions, bounded `free`) is deferred to Phase 2.

Usage

```
read_lgo(path = NULL, text = NULL, as = c("edges", "igraph"))
```

Arguments

`path`	Path to a `.lgo` file (UTF-8 encoded).
`text`	A character scalar (full file text) or character vector (one element per line).
`as`	Output format. Only `"edges"` (canonical edge tibble) is supported in this version; `"igraph"` errors with `legofit_lgo_unsupported` and arrives in Phase 2.

Details

Exactly one of `path` or `text` must be supplied.

Value

A tibble with columns `from`, `to`, `type`, `weight`.

`read_packedancestrymap` *Read genotype data from packedancestrymap files*

Description

Read genotype data from packedancestrymap files

Usage

```
read_packedancestrymap(
  pref,
  inds = NULL,
  pops = NULL,
  first = 1,
  last = Inf,
  transpose = FALSE,
  verbose = TRUE
)
```

Arguments

`pref`	Prefix of the packedancestrymap files
`inds`	Individuals for which data should be read. Defaults to all individuals
`pops`	Populations for which data should be read. Cannot be provided together with 'inds'
`first`	Index of first SNP to read
`last`	Index of last SNP to read
`transpose`	Transpose genotype matrix (default is `snps` x `individuals`)
`verbose`	Print progress updates

Value

A list with the genotype data matrix, the `.ind` file, and the `.snp` file

Examples

```
## Not run:
samples = c('Ind1', 'Ind2', 'Ind3')
geno = read_packedancestrymap(prefix, samples)

## End(Not run)
```

`read_plink` *Read genotype data from* `PLINK` *files*

Description

See genio for a dedicated `R` package for reading and writing `PLINK` files. This function is based on a similar function in the `plink2R` package.

Usage

```
read_plink(pref, inds = NULL, pops = NULL, verbose = FALSE)
```

Arguments

`pref`	Prefix of packedancestrymap files (files have to end in `.geno`, `.ind`, `.snp`)
`inds`	Individuals for which data should be read. Defaults to all individuals
`pops`	Populations for which data should be read. Cannot be provided together with 'inds'
`verbose`	Print progress updates

Value

A list with the genotype data matrix, the `.ind` file, and the `.snp` file

Examples

```
## Not run:
samples = c('Ind1', 'Ind2', 'Ind3')
geno = read_packedancestrymap(prefix, samples)

## End(Not run)
```

`resample_inds` *Run models while leaving out individuals*

Description

These are wrapper functions around various `qp` functions, which will evaluate many models at once. The models are formed by leaving out every individual which belongs to a population with more than one sample, one at a time.

Usage

```
qp3pop_resample_inds(dir, inds, pops, verbose = TRUE, ...)

qpdstat_resample_inds(dir, inds, pops, verbose = TRUE, ...)

qpwave_resample_inds(dir, inds, pops, verbose = TRUE, ...)

qpadm_resample_inds(dir, inds, pops, verbose = TRUE, ...)

qpgraph_resample_inds(dir, inds, pops, verbose = TRUE, ...)
```

Arguments

`dir`	directory with precomputed data
`inds`	vector of individual names
`pops`	vector of population names. Should be the same length as `inds`
`verbose`	print progress updates
`...`	named arguments passed to the `qp` function.

Value

a nested data frame where each model is a row, and the columns are model parameters and model outputs

Examples

```
## Not run:
res = qpadm_resample_inds
unnest(res, weights)

## End(Not run)
```

`resample_snps` *Run models while leaving out SNP blocks*

Description

These are wrapper functions around various `qp` functions, which will evaluate a model on multiple subdivisions of the data. The subdivisions are formed by leaving out one or more SNP block at a time (see `boot` for details).

Usage

```
qp3pop_resample_snps(f2_blocks, boot = FALSE, ...)

qpdstat_resample_snps(f2_blocks, boot = FALSE, ...)

qpwave_resample_snps(f2_blocks, boot = FALSE, ...)

qpadm_resample_snps(f2_blocks, boot = FALSE, ...)

qpgraph_resample_snps(f2_blocks, boot = FALSE, ...)
```

Arguments

`f2_blocks`	a 3d array of blocked f2 statistics
`boot`	If `FALSE` (the default), block-jackknife resampling will be used to compute standard errors. Otherwise, block-bootstrap resampling will be used to compute standard errors. If `boot` is an integer, that number will specify the number of bootstrap resamplings. If `boot = TRUE`, the number of bootstrap resamplings will be equal to the number of SNP blocks.
`...`	named arguments which are passed to the `qp` function.

Value

a nested data frame where each model is a row, and the columns are model parameters and model outputs

Examples

```
## Not run:
res = qpadm_resample_snps(example_f2_blocks, target = target, left = left, right = right)
unnest(res, weights)

## End(Not run)
```

`result_to_json` *Serialize an admixtools result to JSON*

Description

Wraps the output of one of the admixtools fitters (`qpadm()`, `qpwave()`, `qpgraph()`, `qpfstats()`, `f4()`, `f3()`, `f2()`, `qpdstat()`, `qpadm_sweep()`, ...) in a schema-versioned envelope and returns a JSON string (or writes to a file / connection).

Usage

```
result_to_json(
  result,
  fn = NULL,
  args = list(),
  file = "",
  pretty = FALSE,
  digits = NA
)
```

Arguments

`result`	The result list (or single data frame, or array) returned by an admixtools fitter.
`fn`	Character scalar naming the function that produced `result` (e.g. `"qpadm"`, `"qpgraph"`). Stamped into the JSON envelope so consumers can dispatch on the shape. Defaults to `NULL`, which renders as `"unknown"`: useful for ad-hoc serialization where the caller doesn't want to plumb the producer name through. `NA` (any flavour: untyped, `NA_character_`, etc.) is treated the same as `NULL`.
`args`	Optional named list of the arguments used in the call. Echoed verbatim into the envelope's `args` field. Use `list()` (the default) to skip. **Values must be primitives** (scalars, atomic vectors, or nested lists thereof): non-primitives like functions, environments, or external pointers don't have well-defined JSON representations and may serialize to `{}` or error from `jsonlite`.
`file`	Path or connection to write to. The default (`""`) returns the JSON as a character scalar without writing.
`pretty`	If `TRUE`, indent the JSON for human reading. Default `FALSE` (one-line; smaller for streaming).
`digits`	Number of significant digits to use for numeric fields. Default `NA` preserves full precision (matches `jsonlite::toJSON`'s sentinel for "do not round").

Details

Non-R orchestrators currently call admixtools from Rscript and do their own JSON marshaling per result type: five or six bespoke serializer paths, one per output shape. `result_to_json()` consolidates that into a single contract: pass the result, the function name, and (optionally) the args you invoked it with, and get back a stable JSON document. Consumers in any language can `json.load()` and treat admixtools like any other CLI tool.

The envelope is:

```
{
  "schema_version":     1,
  "function":           "qpadm",
  "admixtools_version": "2.0.10",
  "args":               { ... echoed back from the caller ... },
  "result":             { ... whatever the fitter returned ... }
}
```

Tibbles / data.frames are serialized row-wise (each row becomes a JSON object), `NA` becomes `null`, integers/doubles are auto-unboxed, and nested list-columns / lists-of-tibbles are recursed into. 3D arrays (e.g. the raw return of `qpfstats()`) serialize to nested JSON arrays without dimnames; if you need named pop axes, convert to a long tibble first via `cubelyr::as.tbl_cube()` or manual unpacking.

Value

If `file = ""`, a character scalar containing the JSON document. Otherwise the JSON is written to `file` and the path is returned invisibly.

See Also

`qpadm()`, `qpgraph()`, `qpfstats()`

Examples

```
left = c("Altai_Neanderthal.DG", "Vindija.DG")
right = c("Chimp.REF", "Mbuti.DG", "Russia_Ust_Ishim.DG", "Switzerland_Bichon.SG")
target = "Denisova.DG"
fit = qpadm(example_f2_blocks, left, right, target, verbose = FALSE)

# JSON string (one line):
json = result_to_json(fit, fn = "qpadm",
                      args = list(left = left, right = right, target = target))
substr(json, 1, 80)

# Pretty-printed to stdout for inspection:
cat(result_to_json(fit, fn = "qpadm", pretty = TRUE))
```

rotate_models *Rotate populations between left and right*

Description

This functions creates a data frame with population combinations which can be used as the input for qpadm_multi

Usage

```
rotate_models(leftright, target, rightfix = NULL)
```

Arguments

leftright	Populations which will be distributed between left and right
target	Target population
rightfix	Populations which will be on the right side in all models

Value

A data frame with Chi-squared statistics and p-values for each population combination

Examples

```
## Not run:
pops = dimnames(example_f2_blocks)[[1]]
rotate_models(leftright = pops[1:4],
              target = pops[5], rightfix = pops[6:7])

## End(Not run)
```

run_shiny_admixtools *Launch ADMIXTOOLS 2 GUI*

Description

Launch ADMIXTOOLS 2 GUI

Usage

```
run_shiny_admixtools()
```

`satisfies_constraints` *Test constraints on a graph*

Description

This function tests whether a graph satisfies a number of different types of constraints

Usage

```
satisfies_constraints(
  graph,
  nonzero_f4 = NULL,
  admix_constraints = NULL,
  event_order = NULL
)
```

Arguments

`graph`	An admixture graph
`nonzero_f4`	A data frame or matrix with four columns. One row for each f4-statistic which is observed to be significantly non-zero
`admix_constraints`	A data frame with columns `pop`, `min`, `max`
`event_order`	A data frame with columns `earlier1`, `earlier2`, `later1`, `later2`

Value

`TRUE` if all admixture constraints are satisfied, else `FALSE`

See Also

`satisfies_numadmix`, `satisfies_zerof4`, `satisfies_eventorder`

Examples

```
## Not run:
# At least one admixture event for C, and none for D:
constrain_cd = tibble(pop = c('C', 'D'), min = c(1, NA), max = c(NA, 0))
satisfies_numadmix(random_admixturegraph(5, 2), constrain_cd)

## End(Not run)
```

`satisfies_eventorder` *Test f4 constraints on a graph*

Description

This function returns `TRUE` if and only if the admixture graph is compatible with all event orders listed in eventorder

Usage

```
satisfies_eventorder(graph, eventorder, strict = TRUE)
```

Arguments

`graph`	An admixture graph
`eventorder`	A data frame with columns `earlier1`, `earlier2`, `later1`, `later2`, optionally `type`
`strict`	What to do in case some events are not determined by the graph. If `strict = TRUE` (the default), the function will only return `TRUE` if there are no ambiguous constraints. Otherwise, `TRUE` will be returned as long as no constraint is directly contradicted.

Details

Each row in `eventorder` represents a constraint that `earlier1` and `earlier2` split earlier than `later1` and `later2`. If `later2` is `NA`, `later2` will be set to the parent node of `later1`. By default (`type = 1`), a constraint will be satisfied as long as there is any lineage in which a split between `earlier1` and `earlier2` is ancestral to a split between `later1` and `later2`. `type = 2` is stricter and requires that the `earlier1`, `earlier2` split is ancestral to the `later1`, `later2` split in all lineages. In graphs with multiple admixture events there can be multiple splits between `earlier1`, `earlier2` and `later1`, `later2`, and many ways in which these splits can relate to each other. The current implementation only covers some of the many possible topological relationships.

Value

`TRUE` if all constraints are satisfied, else `FALSE`

Examples

```
## Not run:
# Test whether the split between A and B is earlier than the split between C and D,
#   and whether the split between C and D is earlier than the terminal branch leading to E
constrain_events = tribble(
  ~earlier1, ~earlier2, ~later1, ~later2,
  'A', 'B', 'C', 'D',
  'C', 'D', 'E', NA)
satisfies_eventorder(random_admixturegraph(5, 0), eventorder = constrain_events)
```

```
## End(Not run)
```

`satisfies_nonzerof4` *Test f4 constraints on a graph*

Description

This function returns `TRUE` if and only if the admixture graph is compatible with the f4-statistics listed in `nonzero_f4` being non-zero

Usage

```
satisfies_nonzerof4(graph, nonzero_f4)
```

Arguments

`graph`	An admixture graph
`nonzero_f4`	A data frame or matrix with four columns. One row for each f4-statistic which is expected to be non-zero

Value

`TRUE` if all constraints are satisfied, else `FALSE`

Examples

```
## Not run:
# Test whether f4(A,B; C,D) is expected to be non-zero in this graph:
nonzero_f4 = matrix(c('A', 'B', 'C', 'D'), 1)
satisfies_nonzerof4(random_admixturegraph(5, 2), nonzero_f4)

## End(Not run)
```

`satisfies_numadmix` *Test admixture constraints on a graph*

Description

This function returns `TRUE` if and only if the admixture graph satisfies all constraints on the number of admixture events for the populations in `admix_constraints`

Usage

```
satisfies_numadmix(graph, admix_constraints)
```

Arguments

`graph`	An admixture graph
`admix_constraints`	A data frame with columns `pop`, `min`, `max`

Value

`TRUE` if all admixture constraints are satisfied, else `FALSE`

Examples

```
## Not run:
# At least one admixture event for C, and none for D:
constrain_cd = tribble(
  ~pop, ~min, ~max,
  'C', 1, NA,
  'D', NA, 0)
satisfies_numadmix(random_admixturegraph(5, 2), constrain_cd)

## End(Not run)
```

`satisfies_zerof4` *Test f4 constraints on a graph*

Description

This function returns `TRUE` if and only if the admixture graph is compatible with the f4-statistics listed in `nonzero_f4` being non-zero

Usage

```
satisfies_zerof4(graph, zero_f4)
```

Arguments

`graph`	An admixture graph
`zero_f4`	A data frame or matrix with four columns. One row for each f4-statistic which is expected to be zero

Value

`TRUE` if all constraints are satisfied, else `FALSE`

Examples

```
## Not run:
# Test whether Chimp and Altai are a clade with respect to all populations X and Y:
# (whether f4("Chimp", "Altai"; X, Y) is 0 for all pairs of X and Y)
zero_f4 = expand_grid(pop1 = "Chimp", pop2 = "Altai", pop3 = X, pop4 = Y)
satisfies_zerof4(random_admixturegraph(5, 2), zero_f4)

## End(Not run)
```

`set_node_attrs` *Set or update node attributes.*

Description

Set or update node attributes.

Usage

```
set_node_attrs(
  graph,
  name,
  samples = NULL,
  twoN_param = NULL,
  twoN = NULL,
  time_param = NULL,
  time = NULL,
  admix_event_time = NULL
)
```

Arguments

`graph`	An edge tibble.
`name`	Character vector of node names. Must be unique, must exist in the canonical node set.
`samples`	Integer scalar or vector of length(name); NULL leaves unchanged.
`twoN_param`	Character scalar or vector; NULL leaves unchanged.
`twoN`	Numeric scalar or vector; NULL leaves unchanged.
`time_param`	Character scalar or vector; NULL leaves unchanged.
`time`	Numeric scalar or vector; NULL leaves unchanged.
`admix_event_time`	Numeric scalar or vector; NULL leaves unchanged.

Details

Scalar arguments recycle to length(name). Vector arguments must match length(name) exactly. Unknown names error with `admixtools_invalid_graph`. Duplicate names error with `admixtools_invalid_graph`. NA in a scalar or vector sets the cell to NA.

Value

The graph with updated nodes attribute. Refreshes edge time views if `time` or `admix_event_time` was set.

`shortest_unique_prefixes` *Return shortest unique prefixes*

Description

Return shortest unique prefixes

Usage

```
shortest_unique_prefixes(strings, min_length = 1)
```

Arguments

`strings`	A character vector
`min_length`	Minimum length of prefixes

Value

Character vector with shortest unique prefixes

Examples

```
strings = c("Abra", "Abracadabra", "Simsalabim")
shortest_unique_prefixes(strings)
```

`simplify_graph` *Remove redundant edges*

Description

Nodes with in and out degree of one will be removed.

Usage

```
simplify_graph(graph)
```

Arguments

`graph`	an igraph object

Examples

```
simple = simplify_graph(example_igraph)
plot_graph(example_igraph)
plot_graph(simple)
```

`split_mat` *Split a matrix into blocks*

Description

This function splits a large matrix into smaller blocks with `cols_per_chunk` columns per block, and saves them as `.rds` files with prefix `prefix`

Usage

```
split_mat(mat, cols_per_chunk, prefix, overwrite = TRUE, verbose = TRUE)
```

Arguments

`mat`	The matrix to be split
`cols_per_chunk`	Number of columns per block
`prefix`	Prefix of output files
`overwrite`	Overwrite existing files (default `TRUE`)
`verbose`	Print progress updates

See Also

`packedancestrymap_to_afs`, `afs_to_f2`

Examples

```
## Not run:
afdat = packedancestrymap_to_afs('path/to/packedancestrymap_prefix', allpopulations)
split_mat(afdat$afs, cols_per_chunk = 20, prefix = 'afdat_split_v42.1/afs')

## End(Not run)
```

`split_multifurcations` *Split nodes with more than two edges*

Description

Split nodes with more than two edges

Usage

```
split_multifurcations(graph)
```

Arguments

`graph`	An admixture graph

Value

A new admixture graph in which no node has more than two incoming or outgoing edges

---

`spr_all` *Modify a graph by regrafting a subcomponent*

Description

Modify a graph by regrafting a subcomponent

Usage

```
spr_all(graph, fix_outgroup = TRUE)
```

Arguments

`graph`	An admixture graph
`fix_outgroup`	Prevent the outgroup from being regrafted (default `TRUE`).

Value

A new admixture graph

`spr_leaves` *Modify a graph by regrafting a leaf*

Description

Modify a graph by regrafting a leaf

Usage

```
spr_leaves(graph, fix_outgroup = TRUE)
```

Arguments

`graph`	An admixture graph
`fix_outgroup`	Prevent the outgroup leaf from being regrafted (default `TRUE`).

Value

A new admixture graph

`summarize_descendants`
List leaf nodes for all internal nodes

Description

List leaf nodes for all internal nodes

Usage

```
summarize_descendants(graph)
```

Arguments

`graph`	An admixture graphs

Value

A data frame with columns `from`, `to`, `id`

Examples

```
## Not run:
summarize_descendants(graph)

## End(Not run)
```

summarize_descendants_list *List leaf nodes for all internal nodes in a list of graphs*

Description

List leaf nodes for all internal nodes in a list of graphs

Usage

```
summarize_descendants_list(graphlist, rename = FALSE)
```

Arguments

graphlist	A list of admixture graphs
rename	If FALSE (default), the output will be a data frame indicating how often each node in each graph is observed in all other graphs. If TRUE, the output will be a list, where the inner nodes will be renamed to have these percentages as part of their name. plot_graph will print the percentages of graphs renamed in this way.

Value

A data frame with columns graph, from, n, frac

Examples

```
## Not run:
summarize_descendants_list(graphlist)

## End(Not run)
```

summarize_eventorder *List population split events in a graph*

Description

List population split events in a graph

Usage

```
summarize_eventorder(graph, unique_only = TRUE)
```

Arguments

graph	An admixture graph
unique_only	Return only unique, unambiguous event-order relationships (default TRUE).

Value

A data frame with columns `earlier1`, `earlier2`, `later1`, `later2`

Examples

```
## Not run:
summarize_eventorder(example_igraph)

## End(Not run)
```

---

summarize_eventorder_list

List population split events in a list of graphs

Description

List population split events in a list of graphs

Usage

```
summarize_eventorder_list(graphlist)
```

Arguments

`graphlist`	A list of admixture graphs

Value

A data frame with columns `earlier1`, `earlier2`, `later1`, `later2`, `n`, `frac`

Examples

```
## Not run:
summarize_eventorder(graphlist)

## End(Not run)
```

summarize_fits *Summarize graph fits*

Description

This function takes the output of qpgraph_resample_snps and creates a data frame with summaries of the estimated graph parameters. `weight` has the mean of all fits, while `low` and `high` have lower and upper quantiles.

Usage

```
summarize_fits(fits, q_low = 0.05, q_high = 0.95)
```

Arguments

`fits`	A nested data frame where each line is the output of `qpgraph()`
`q_low`	Lower quantile for the weight confidence interval (default 0.05).
`q_high`	Upper quantile for the weight confidence interval (default 0.95).

Value

A data frame summarizing the fits

See Also

qpgraph_resample_snps

summarize_numadmix *List number of admixture events for each population*

Description

List number of admixture events for each population

Usage

```
summarize_numadmix(graph)
```

Arguments

`graph`	An admixture graph

Value

A data frame with columns `pop` and `nadmix`

Examples

```
## Not run:
summarize_numadmix(example_igraph)

## End(Not run)
```

`summarize_numadmix_list` *List number of admixture events for each population in a list of graphs*

Description

List number of admixture events for each population in a list of graphs

Usage

```
summarize_numadmix_list(graphlist)
```

Arguments

`graphlist`	A list of admixture graphs

Value

A data frame with columns `pop`, `mean`, `mean_nonzero`, `min`, `max`

Examples

```
## Not run:
summarize_numadmix_list(graphlist)

## End(Not run)
```

`summarize_proxies` *Assign proxy populations to admixed populations*

Description

Assign proxy populations to admixed populations

Usage

```
summarize_proxies(graph)
```

Arguments

`graph`	An admixture graph in igraph format, or as edge list with column `weight`

Value

A data frame with columns `pop`, `nadmix`, `proxy`, `nproxies` (and `weight`)

Examples

```
## Not run:
summarize_proxies(example_igraph)

## End(Not run)
```

`summarize_proxies_list`

List proxy populations in graphlist

Description

List proxy populations in graphlist

Usage

```
summarize_proxies_list(graphlist)
```

Arguments

`graphlist`	A list of admixture graphs

Value

A data frame with columns `pop`, `proxy`, `nadmix`, `nproxies`, `n`, `frac`

Examples

```
## Not run:
summarize_proxies_list(graphlist)

## End(Not run)
```

summarize_triples *Summarize triples across graphs*

Description

This function summarizes topologies of population triples across graphs.

Usage

```
summarize_triples(graphs)
```

Arguments

graphs	A list of graphs

Value

A data frame with one line for each population triple and columns summarizing the relationship of each triple across graphs.

- clade12: Fraction of graphs in which pop1 and pop2 form a clade
- x12: Fraction of graphs in which pop1 admixes into pop2 at the exclusion of pop3
- toptopo: A binary string representation of the most common topology. Digits represent x12, x21, x13, x31, x23, x32
- toptopocnt: The number of graphs in which toptopo was observed
- topos: The counts of all topologies

summarize_zerof4 *List clades in a graph*

Description

List clades in a graph

Usage

```
summarize_zerof4(graph, eps = 1e-07)
```

Arguments

graph	An admixture graph
eps	Used to determine what counts as zero

Value

A data frame with all (non-redundant) zero f4-statistics

Examples

```
## Not run:
summarize_zerof4(example_igraph)

## End(Not run)
```

`summarize_zerof4_list` *List clades in a list of graphs*

Description

List clades in a list of graphs

Usage

```
summarize_zerof4_list(graphlist)
```

Arguments

`graphlist`	A list of admixture graphs

Value

A data frame with columns `pop1`, `pop2`, `pop3`, `pop4`, `n`, `frac`

Examples

```
## Not run:
summarize_zerof4_list(graphlist)

## End(Not run)
```

`swap_leaves` *Modify a graph by swapping two leaf nodes*

Description

Modify a graph by swapping two leaf nodes

Usage

```
swap_leaves(graph, fix_outgroup = TRUE)
```

Arguments

`graph`	An admixture graph
`fix_outgroup`	Keep outgroup in place

Value

A new admixture graph

`test_cladality` *Test if two sets of populations form two clades*

Description

A thin wrapper around `qpadm_p` with `rnk` set to zero

Usage

```
test_cladality(f2_data, left, right, fudge = 1e-04, boot = FALSE, cpp = TRUE)
```

Arguments

`f2_data`	Blocked f2-statistics (3d array), a directory path, or a genotype file prefix.
`left`	Left populations (sources)
`right`	Right populations (outgroups)
`fudge`	Value added to diagonal matrix elements before inverting
`boot`	If `FALSE` (the default), block-jackknife resampling will be used to compute standard errors. Otherwise, block-bootstrap resampling will be used to compute standard errors. If `boot` is an integer, that number will specify the number of bootstrap resamplings. If `boot = TRUE`, the number of bootstrap resamplings will be equal to the number of SNP blocks.
`cpp`	Use C++ functions. Setting this to `FALSE` will be slower but can help with debugging.

`tree_in_graph` *Test if a tree is part of a graph*

Description

This function tests whether a tree is part of a graph. This is useful for testing whether a Y-chromosome tree is consistent with an autosomal admixture graph. Leaf node names matter, but internal node names are ignored.

Usage

```
tree_in_graph(tree, graph)
```

Arguments

`tree`	An admixture graph without admixture event
`graph`	An admixture graph

Value

`TRUE` if all admixture constraints are satisfied, else `FALSE`

Examples

```
## Not run:
tree = graph_splittrees(example_igraph) %>% pull(graph) %>% pluck(1)
tree_in_graph(tree, example_igraph)

## End(Not run)
```

`tree_neighbors` *Find all trees within SPR distance of 1*

Description

Returns all trees which can be reached through one iteration of subtree-prune-and-regraft

Usage

```
tree_neighbors(tree)
```

Arguments

`tree`	A tree in `igraph` format

Value

A data frame with all trees

`unidentifiable_edges` *Find all unidentifiable edges*

Description

This function generates and evaluates admixture graphs in `numgen` iterations to find well fitting admixturegraphs.

Usage

```
unidentifiable_edges(graph)
```

Arguments

`graph`	An admixture graph

Value

A data frame with all unidentifiable graph parameters

`write_dot` *Convert graph to dot format*

Description

Convert graph to dot format

Usage

```
write_dot(
  graph,
  outfile = stdout(),
  size1 = 7.5,
  size2 = 10,
  title = "",
  dot2pdf = FALSE,
  fontsize = 14,
  color = TRUE,
  hide_weights = FALSE,
  highlight_unidentifiable = FALSE,
  nodesep = 0.25,
  ranksep = 0.5,
  fix_names = FALSE
)
```

Arguments

`graph`	Graph as igraph object or edge list (columns labelled 'from', 'to', 'weight')
`outfile`	Output file name
`size1`	Maximum width of the rendered graph in inches (the first value of Graphviz's `size` attribute). The default is `7.5`.
`size2`	Maximum height of the rendered graph in inches (the second value of Graphviz's `size` attribute). The default is `10`.
`title`	Title displayed above the graph. Defaults to the empty string.
`dot2pdf`	If `FALSE`, the function will terminate after writing the dot file. If `TRUE`, it will try to execute the `dot -Tpdf` comment to create pdf file.
`fontsize`	Font size of edge and node labels
`color`	A boolean specifying if the plot will be in color or grey scale.
`hide_weights`	A boolean value specifying if the drift values on the edges will be hidden. The default is `FALSE`.
`highlight_unidentifiable`	Highlight unidentifiable edges in red. Can be slow for large graphs.
`nodesep`	The minimum space between two adjacent nodes in the same rank, in inches. The default is `0.25`.

ranksep	Sets the rank separation, in inches. This is the minimum vertical distance between the bottom of the nodes in one rank and the tops of nodes in the next. The default is 0.5.
fix_names	If TRUE, replaces the dots (.) and dashes (-) in population names with underscores. The default is FALSE.

Examples

```
## Not run:
results = qpgraph(example_f2_blocks, example_graph)
write_dot(results$edges)

## End(Not run)
```

---

write_f2 *Write blocked f2 estimates to disk*

Description

This function takes 3d arrays of blocked f2, allele frequency products, and counts, splits them by population pair, and writes each pair to a separate .rds file under {outdir}/{pop1}/{pop2}.rds.

Usage

```
write_f2(est_arr, count_arr, outdir, id = "f2", overwrite = FALSE)
```

Arguments

est_arr	A 3d array with blocked f2, allele frequency products, or fst estimates for each population pair. The first two dimensions of each array have to have population names.
count_arr	A 3d array of the same dimension with counts
outdir	Directory where data will be stored
id	Postfix showing the type of statistic (”f2”, ”ap”, or ”fst”)
overwrite	Overwrite existing files in outdir

See Also

read_f2

Examples

```
## Not run:
write_f2(f2_arr, count_arr, outdir = 'path/to/f2stats/')

## End(Not run)
```

Index