Package 'bnpsd'

May 15, 2019

Title Simulate Genotypes from the BN-PSD Admixture Model

Version 1.1.1

Description The Pritchard-Stephens-Donnelly (PSD) admixture model has k intermediate subpopulations from which n individuals draw their alleles dictated by their individual-specific admixture proportions. The BN-PSD model additionally imposes the Balding-Nichols (BN) allele frequency model to the intermediate populations, which therefore evolved independently from a common ancestral population T with subpopulation-specific FST (Wright's fixation index) parameters. The BN-PSD model can be used to yield complex population structures. Method described in Ochoa and Storey (2016) <doi:10.1101/083923>.

Depends

Imports stats

Suggests popkin (>= 1.2.2), testthat, knitr, rmarkdown, RColorBrewer

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1.9000

VignetteBuilder knitr

URL https://github.com/StoreyLab/bnpsd/

BugReports https://github.com/StoreyLab/bnpsd/issues

NeedsCompilation no

Author Alejandro Ochoa [aut, cre] (<https://orcid.org/0000-0003-4928-3403>), John D. Storey [aut] (<https://orcid.org/0000-0001-5992-402X>)

Maintainer Alejandro Ochoa <alejandro.ochoa@duke.edu>

Repository CRAN

Date/Publication 2019-05-15 17:40:03 UTC

R topics documented:

`admix_prop_1d_circular`

Construct admixture proportion matrix for circular 1D geography

---

Description

Assumes k intermediate subpopulations placed along a circumference (the $[0, 2\pi]$ line that wraps around) with even spacing spread by random walks (see details below), then n individuals sampled equally spaced in $[a, b]$ (default $[0, 2\pi]$ with a small gap so first and last individual do not overlap) draw their admixture proportions relative to the Von Mises density that models the random walks of each of these intermediate subpopulations. The spread of the random walks (the $\sigma = 1/\sqrt{\kappa}$ of the Von Mises densities) is set to `sigma` if not missing, otherwise σ is found numerically to give the desired bias coefficient `bias_coeff`, the coancestry matrix of the intermediate subpopulations `coanc_subpops` (up to a scalar factor), and the final F_{ST} of the admixed individuals (see details below).

Usage

```
admix_prop_1d_circular(n_ind, k_subpops, sigma = NA,
  coord_ind_first = 2 * pi/(2 * n_ind), coord_ind_last = 2 * pi * (1 -
  1/(2 * n_ind)), bias_coeff, coanc_subpops, fst)
```

Arguments

`n_ind`	Number of individuals
`k_subpops`	Number of intermediate subpopulations
`sigma`	Spread of intermediate subpopulations (approximate standard deviation of Von Mises densities, see above) The edge cases `sigma = 0` and `sigma = Inf` are handled appropriately!

`coord_ind_first`
: Location of first individual

`coord_ind_last`
: Location of last individual

OPTIONS FOR BIAS COEFFICIENT VERSION

`bias_coeff`
: The desired bias coefficient, which specifies σ indirectly. Required if `sigma` is missing.

`coanc_subpops`
: The length-k vector of inbreeding coefficients (or F_{ST}'s) of the intermediate subpopulations, up to a scaling factor (which cancels out in calculations). Required if `sigma` is missing.

`fst`
: The desired final F_{ST} of the admixed individuals. Required if `sigma` is missing.

Details

Assuming the full range of $[0, 2\pi]$ is considered, and the first and last individuals do not overlap, the gap between individuals is $\Delta = 2\pi/n$. To not have any individuals on the edge, we place the first individual at $\Delta/2$ and the last at $2\pi - \Delta/2$. The location of subpopulation j is

$$\Delta/2 + (j - 1/2)/k(2\pi - \Delta),$$

chosen to agree with the default correspondence between individuals and subpopulations of the linear 1D geography admixture scenario (`admix_prop_1d_linear`).

When `sigma` is missing, the function determines its value using the desired `bias_coeff`, `coanc_subpops` up to a scalar factor, and `fst`. Uniform weights for the final generalized F_{ST} are assumed. The scaling factor of the input `coanc_subpops` is irrelevant because it cancels out in `bias_coeff`; after `sigma` is found, `coanc_subpops` is rescaled to give the desired final F_{ST}. However, the function stops with a fatal error if the rescaled `coanc_subpops` takes on any values greater than 1, which are not allowed since `coanc_subpops` are IBD probabilities.

Value

If `sigma` was provided, the $n \times k$ admixture proportion matrix. If `sigma` is missing, a named list is returned containing `admix_proportions`, the rescaled `coanc_subpops`, and the `sigma` that together give the desired $bias_coeff$ and final F_{ST} of the admixed individuals.

Examples

```
# admixture matrix for 1000 individuals drawing alleles from 10 subpops
# and a spread of about 2 standard deviations along the circular 1D geography
admix_proportions <- admix_prop_1d_circular(n_ind = 1000, k_subpops = 10, sigma = 2)

# a similar model but with a bias coefficient of exactly 1/2
k_subpops <- 10
# FST vector for intermediate independent subpops, up to a factor (will be rescaled below)
coanc_subpops <- 1 : k_subpops
obj <- admix_prop_1d_circular(
    n_ind = 1000,
    k_subpops = k_subpops,
    bias_coeff = 0.5,
    coanc_subpops = coanc_subpops,
```

```
    fst = 0.1 # desired final FST of admixed individuals
)

# in this case return value is a named list with three items:
admix_proportions <- obj$admix_proportions

# rescaled coancestry data (matrix or vector) for intermediate subpops
coanc_subpops <- obj$coanc_subpops

# and the sigma that gives the desired bias_coeff and final FST
sigma <- obj$sigma
```

`admix_prop_1d_linear` *Construct admixture proportion matrix for 1D geography*

Description

Assumes k intermediate subpopulations placed along a line at locations $1 : k$ spread by random walks, then n individuals sampled equally spaced in $[a, b]$ (default $[0.5, k + 0.5]$) draw their admixture proportions relative to the Normal density that models the random walks of each of these intermediate subpopulations. The spread of the random walks (the σ of the Normal densities) is set to `sigma` if not missing, otherwise σ is found numerically to give the desired bias coefficient `bias_coeff`, the coancestry matrix of the intermediate subpopulations `coanc_subpops` (up to a scalar factor), and the final F_{ST} of the admixed individuals (see details below).

Usage

```
admix_prop_1d_linear(n_ind, k_subpops, sigma = NA,
  coord_ind_first = 0.5, coord_ind_last = k_subpops + 0.5, bias_coeff,
  coanc_subpops, fst)
```

Arguments

`n_ind`	Number of individuals.
`k_subpops`	Number of intermediate subpopulations.
`sigma`	Spread of intermediate subpopulations (standard deviation of normal densities). The edge cases `sigma = 0` and `sigma = Inf` are handled appropriately!
`coord_ind_first`	Location of first individual.
`coord_ind_last`	Location of last individual.
	OPTIONS FOR BIAS COEFFICIENT VERSION
`bias_coeff`	The desired bias coefficient, which specifies σ indirectly. Required if `sigma` is missing.
`coanc_subpops`	The $k \times k$ coancestry matrix of the intermediate subpopulations, or equivalent vector or scalar forms (which model independent subpopulations), up to a scaling factor (which cancels out in calculations) Required if `sigma` is missing.
`fst`	The desired final F_{ST} of the admixed individuals. Required if `sigma` is missing.

Details

When sigma is missing, the function determines its value using the desired bias_coeff, coanc_subpops up to a scalar factor, and fst. Uniform weights for the final generalized F_{ST} are assumed. The scale of coanc_subpops is irrelevant because it cancels out in bias_coeff; after sigma is found, coanc_subpops is rescaled to give the desired final F_{ST}. However, the function stops with a fatal error if the rescaled coanc_subpops takes on any values greater than 1, which are not allowed since coanc_subpops are IBD probabilities.

Value

If sigma was provided, the $n \times k$ admixture proportion matrix. If sigma is missing, a named list is returned containing admix_proportions, the rescaled coanc_subpops, and the sigma that together give the desired $bias_coeff$ and final F_{ST} of the admixed individuals.

Examples

```
# admixture matrix for 1000 individuals drawing alleles from 10 subpops
# and a spread of 2 standard deviations along the 1D geography
admix_proportions <- admix_prop_1d_linear(n_ind = 1000, k_subpops = 10, sigma = 2)

# as sigma approaches zero, admix_proportions approaches the independent subpopulations matrix
admix_prop_1d_linear(n_ind = 10, k_subpops = 2, sigma = 0)

# a similar model but with a bias coefficient of exactly 1/2
k_subpops <- 10
# FST vector for intermediate independent subpops, up to a factor (will be rescaled below)
coanc_subpops <- 1 : k_subpops
obj <- admix_prop_1d_linear(
    n_ind = 1000,
    k_subpops = k_subpops,
    bias_coeff = 0.5,
    coanc_subpops = coanc_subpops,
    fst = 0.1 # desired final FST of admixed individuals
)

# in this case return value is a named list with three items:
# admixture proportions
admix_proportions <- obj$admix_proportions

# rescaled coancestry data (matrix or vector) for intermediate subpops
coanc_subpops <- obj$coanc_subpops

# and the sigma that gives the desired bias_coeff and final FST
sigma <- obj$sigma
```

---

admix_prop_indep_subpops *Construct admixture proportion matrix for independent subpopulations*

Description

This function constructs an admixture proportion matrix where every individuals is actually unadmixed (draws its full ancestry from a single intermediate subpopulation). The inputs are the vector of subpopulation labels `labs` for every individual (length n), and the length-k vector of unique subpopulations `subpops` in the desired order. If `subpops` is missing, the sorted unique subpopulations observed in `labs` is used. This function returns the admixture proportion matrix, for each individual 1 for the column corresponding to its subpopulation, 0 otherwise.

Usage

```
admix_prop_indep_subpops(labs, subpops)
```

Arguments

`labs`	Length-n vector of subpopulation labels
`subpops`	Optional length-k vector of unique subpopulations in desired order. Stops if `subpops` does not contain all unique labels in `labs` (no error if `subpops` contains additional labels).

Value

The $n \times k$ admixture proportion matrix. The unique subpopulation labels are given in the column names.

Examples

```
# vector of subpopulation memberships
labs <- c(1, 1, 1, 2, 2, 3, 1)
# admixture matrix with subpopulations (along columns) sorted
admix_proportions <- admix_prop_indep_subpops(labs)

# declare subpopulations in custom order
subpops <- c(3, 1, 2)
# columns will be reordered to match subpops as provided
admix_proportions <- admix_prop_indep_subpops(labs, subpops)

# declare subpopulations with unobserved labels
subpops <- 1:5
# note columns 4 and 5 will be false for all individuals
admix_proportions <- admix_prop_indep_subpops(labs, subpops)
```

---

bnpsd *A package for modeling and simulating an admixed population*

Description

The underlying model is called the BN-PSD admixture model, which combines the Balding-Nichols (BN) allele frequency model for the intermediate subpopulations with the Pritchard-Stephens-Donnelly (PSD) model of individual-specific admixture proportions. The BN-PSD model enables the simulation of complex population structures, ideal for illustrating challenges in kinship coefficient and F_{ST} estimation. Simulated loci are drawn independently (in linkage equilibrium).

Author(s)

Maintainer: Alejandro Ochoa <alejandro.ochoa@duke.edu> (0000-0003-4928-3403)

Authors:

- John D. Storey <jstorey@princeton.edu> (0000-0001-5992-402X)

See Also

Useful links:

- https://github.com/StoreyLab/bnpsd/
- Report bugs at https://github.com/StoreyLab/bnpsd/issues

Examples

```
# dimensions of data/model
# number of loci
m_loci <- 10
# number of individuals
n_ind <- 5
# number of intermediate subpops
k_subpops <- 2

# define population structure
# FST values for k = 2 subpopulations
inbr_subpops <- c(0.1, 0.3)
# admixture proportions from 1D geography
admix_proportions <- admix_prop_1d_linear(n_ind, k_subpops, sigma = 1)
# also available:
# - admix_prop_1d_circular
# - admix_prop_indep_subpops

# get pop structure parameters of the admixed individuals
# the coancestry matrix
coancestry <- coanc_admix(admix_proportions, inbr_subpops)
# FST of admixed individuals
Fst <- fst_admix(admix_proportions, inbr_subpops)

# draw all random allele freqs and genotypes
out <- draw_all_admix(admix_proportions, inbr_subpops, m_loci)
# genotypes
X <- out$X
# ancestral allele frequencies (AFs)
```

```
p_anc <- out$p_anc

# OR... draw each vector or matrix separately
# provided for additional flexibility
# ancestral AFs
p_anc <- draw_p_anc(m_loci)
# independent subpops (intermediate) AFs
p_subpops <- draw_p_subpops(p_anc, inbr_subpops)
# individual-specific AFs
p_ind <- make_p_ind_admix(p_subpops, admix_proportions)
# genotypes
X <- draw_genotypes_admix(p_ind)
```

`coanc_admix` *Construct the coancestry matrix of an admixture model*

Description

In the most general case, the $n \times n$ coancestry matrix Θ of admixed individuals is determined by the $n \times k$ admixture proportion matrix Q and the $k \times k$ intermediate subpopulation coancestry matrix Ψ, given by

$$\Theta = Q\Psi Q^T$$

In the BN-PSD model Ψ is a diagonal matrix (with F_{ST} values for the intermediate subpopulations along the diagonal, zero values off-diagonal).

Usage

```
coanc_admix(admix_proportions, coanc_subpops)
```

Arguments

`admix_proportions`	The $n \times k$ admixture proportion matrix
`coanc_subpops`	Either the $k \times k$ intermediate subpopulation coancestry matrix (for the complete admixture model), or the length-k vector of intermediate subpopulation F_{ST} values (for the BN-PSD model), or a scalar F_{ST} value shared by all intermediate subpopulations.

Value

The $n \times n$ coancestry matrix.

Examples

```
# a trivial case: unadmixed individuals from independent subpopulations
# number of individuals and subpops
n_ind <- 5
# unadmixed individuals
admix_proportions <- diag(rep.int(1, n_ind))
# equal Fst for all subpops
coanc_subpops <- 0.2
# diagonal coancestry matryx
coancestry <- coanc_admix(admix_proportions, coanc_subpops)

# a more complicated admixture model
# number of individuals
n_ind <- 5
# number of intermediate subpops
k_subpops <- 2
# non-trivial admixture proportions
admix_proportions <- admix_prop_1d_linear(n_ind, k_subpops, sigma = 1)
# different Fst for each of the k_subpops
coanc_subpops <- c(0.1, 0.3)
# non-trivial coancestry matrix
coancestry <- coanc_admix(admix_proportions, coanc_subpops)
```

---

coanc_to_kinship *Transform coancestry matrix to kinship matrix*

Description

Let $\Theta = (\theta_{jk})$ be the coancestry matrix and $\Phi = (\varphi_{jk})$ be the kinship matrix. These matrices agree off-diagonal, but the diagonal gets transformed as

$$\phi_{jj} = \frac{1 + \theta_{jj}}{2}.$$

Below n is the number of individuals.

Usage

```
coanc_to_kinship(coancestry)
```

Arguments

coancestry	The $n \times n$ coancestry matrix

Value

The $n \times n$ kinship matrix, preserving column and row names.

See Also

The inverse function is given by inbr_diag.

Examples

```
# a trivial case: unadmixed individuals from independent subpopulations
# number of individuals/subpops
n_ind <- 5
# unadmixed individuals
admix_proportions <- diag(rep.int(1, n_ind))
# equal Fst for all subpops
inbr_subpops <- 0.2
# diagonal coancestry matryx
coancestry <- coanc_admix(admix_proportions, inbr_subpops)
kinship <- coanc_to_kinship(coancestry)

# a more complicated admixture model
# number of individuals
n_ind <- 5
# number of intermediate subpops
k_subpops <- 2
# non-trivial admixture proportions
admix_proportions <- admix_prop_1d_linear(n_ind, k_subpops, sigma = 1)
# different Fst for each of the k subpops
inbr_subpops <- c(0.1, 0.3)
# non-trivial coancestry matrix
coancestry <- coanc_admix(admix_proportions, inbr_subpops)
kinship <- coanc_to_kinship( coancestry )
```

---

draw_all_admix *Simulate random allele frequencies and genotypes from the BN-PSD admixture model*

Description

This function returns simulated ancestral, intermediate, and individual-specific allele frequencies and genotypes given the admixture structure, as determined by the admixture proportions and the vector of intermediate subpopulation F_{ST} values. The function is a wrapper around draw_p_anc, draw_p_subpops, make_p_ind_admix, and draw_genotypes_admix with additional features such as requiring polymorphic loci. Importantly, by default fixed loci are re-drawn from the start (starting from the ancestral allele frequencies) so no fixed loci are in the output and no biases are introduced by re-drawing genotypes conditional on any of the previous allele frequencies (ancestral, intermediate, or individual-specific). Below m is the number of loci, n is the number of individuals, and k is the number of intermediate subpopulations.

Usage

```
draw_all_admix(admix_proportions, inbr_subpops, m_loci,
  want_genotypes = TRUE, want_p_ind = FALSE, want_p_subpops = FALSE,
  want_p_anc = TRUE, low_mem = FALSE, verbose = FALSE,
  require_polymorphic_loci = TRUE)
```

Arguments

admix_proportions	The $n \times k$ matrix of admixture proportions.
inbr_subpops	The length-k vector (or scalar) of intermediate subpopulation F_{ST} values.
m_loci	The number of loci to draw.
want_genotypes	If TRUE (default), includes the matrix of random genotypes in the return list.
want_p_ind	If TRUE (NOT default), includes the matrix of individual-specific allele frequencies in the return list.
want_p_subpops	If TRUE (NOT default), includes the matrix of random intermediate subpopulation allele frequencies in the return list.
want_p_anc	If TRUE (default), includes the matrix of random ancestral allele frequencies in the return list.
low_mem	If TRUE, uses a low-memory algorithm to raw genotypes without storing or returning the corresponding 'p_ind' matrix.
verbose	If TRUE, prints messages for every stage in the algorithm.
require_polymorphic_loci	If TRUE (default), returned genotype matrix will not include any fixed loci (loci that happened to be fixed are drawn again, starting from their ancestral allele frequencies, and checked iteratively until no fixed loci remain, so that the final number of polymorphic loci is exactly $m_{l}oci$).

Value

A named list that includes the following randomly-generated data in this order:

X: An $m \times n$ matrix of genotypes. Included if `want_genotypes = TRUE`.

p_anc: A length-m vector of ancestral allele frequencies. Included if `want_p_anc = TRUE`.

p_subpops: An $m \times k$ matrix of intermediate subpopulation allele frequencies Included if `want_p_subpops = TRUE`.

p_ind: An $m \times n$ matrix of individual-specific allele frequencies. Included only if both `want_p_ind = TRUE` and `low_mem = FALSE`.

Examples

```
# dimensions
# number of loci
m_loci <- 10
# number of individuals
n_ind <- 5
# number of intermediate subpops
```

```
k_subpops <- 2

# define population structure
# FST values for k = 2 subpopulations
inbr_subpops <- c(0.1, 0.3)
# admixture proportions from 1D geography
admix_proportions <- admix_prop_1d_linear(n_ind, k_subpops, sigma = 1)

# draw all random allele freqs and genotypes
out <- draw_all_admix(admix_proportions, inbr_subpops, m_loci)

# return value is a list with these items:

# genotypes
X <- out$X

# ancestral AFs
p_anc <- out$p_anc

# # these are excluded by default, but would be included if ...
# # ... `want_p_subpops == TRUE`
# # intermediate subpopulation AFs
# p_subpops <- out$p_subpops
#
# # ... `want_p_ind == TRUE` and `low_mem = FALSE`
# # individual-specific AFs
# p_ind <- out$p_ind
```

`draw_genotypes_admix` *Draw genotypes from the admixture model*

Description

Given the Individual-specific Allele Frequency (IAF) π_{ij} for locus i and individual j, genotypes are drawn binomially:

$$x_{ij}|\pi_{ij} \sim \text{Binomial}(2, \pi_{ij}).$$

Below m is the number of loci, n the number of individuals, and k the number of intermediate subpopulations. If an admixture proportion matrix Q is provided as the second argument, the first argument P is treated as the intermediate subpopulation allele frequency matrix and the IAF matrix is given by

$$PQ^T.$$

If Q is missing, then P is treated as the IAF matrix.

Usage

```
draw_genotypes_admix(p_ind, admix_proportions = NULL, low_mem = FALSE)
```

Arguments

p_ind	The $m \times n$ IAF matrix (if admix_proportions is missing) or the $m \times k$ intermediate subpopulation allele frequency matrix (if admix_proportions is present)
admix_proportions	The optional $n \times k$ admixture proportion matrix
low_mem	If TRUE, the low-memory algorithm is used (admix_proportions must be present)

Details

To reduce memory, set low_mem = TRUE to draw genotypes one locus at the time from P and Q (both must be present). This low-memory algorithm prevents the construction of the entire IAF matrix, but is considerably slower than the standard algorithm.

Value

The $m \times n$ genotype matrix

Examples

```
# dimensions
# number of loci
m_loci <- 10
# number of individuals
n_ind <- 5
# number of intermediate subpops
k_subpops <- 2

# define population structure
# FST values for k = 2 subpops
inbr_subpops <- c(0.1, 0.3)
# non-trivial admixture proportions
admix_proportions <- admix_prop_1d_linear(n_ind, k_subpops, sigma = 1)

# draw allele frequencies
# vector of ancestral allele frequencies
p_anc <- draw_p_anc(m_loci)

# matrix of intermediate subpop allele freqs
p_subpops <- draw_p_subpops(p_anc, inbr_subpops)

# matrix of individual-specific allele frequencies
p_ind <- make_p_ind_admix(p_subpops, admix_proportions)

# draw genotypes from intermediate subpops (one individual each)
X_subpops <- draw_genotypes_admix(p_subpops)

# and genotypes for admixed individuals
X_ind <- draw_genotypes_admix(p_ind)

# draw genotypes for admixed individuals without p_ind intermediate
# (p_ind is computer internally and discarded when done)
```

```
X_ind <- draw_genotypes_admix(p_subpops, admix_proportions)

# use low-memory version (p_ind is computed by row, never fully in memory)
X_ind <- draw_genotypes_admix(p_subpops, admix_proportions, low_mem = TRUE)
```

---

draw_p_anc *Draw random uniform ancestral allele frequencies*

Description

This is simply a wrapper around `runif` with different defaults and additional validations.

Usage

```
draw_p_anc(m_loci, p_min = 0.01, p_max = 0.5)
```

Arguments

m_loci	Number of loci to draw
p_min	Minimum allele frequency to draw
p_max	Maximum allele frequency to draw

Value

A length-m vector of random ancestral allele frequencies

Examples

```
p_anc <- draw_p_anc(m_loci = 10)
```

---

draw_p_subpops *Draw allele frequencies for independent subpopulations*

Description

Allele frequencies $p_i^{S_u}$ for independent subpopulations S_u at locus i are drawn from the Balding-Nichols distribution with ancestral allele frequency p_i^T and F_{ST} parameter $f_{S_u}^T$ as

$$p_i^{S_u} \sim \text{Beta}(\nu_u p_i^T, \nu_u(1 - p_i^T)),$$

where $\nu_u = 1/f_{S_u}^T - 1$. Below m is the number of loci and k is the number of subpopulations.

Usage

```
draw_p_subpops(p_anc, inbr_subpops, m_loci = NA, k_subpops = NA)
```

Arguments

p_anc	The length-m vector of ancestral allele frequencies per locus.
inbr_subpops	The length-k vector of subpopulation F_{ST} values.
m_loci	Optional. The desired number of loci m, to be used if p_anc is a scalar. Stops if both length(p_anc) > 1 and m_loci are set and they disagree.
k_subpops	Optional. The desired number of subpopulations k, to be used if inbr_subpops is a scalar. Stops if both length(inbr_subpops) > 1 and k_subpops are set and they disagree.

Value

The $m \times k$ matrix of independent subpopulation allele frequencies

Examples

```
# a typical, non-trivial example
# number of loci
m_loci <- 10
# random vector of ancestral allele frequencies
p_anc <- draw_p_anc(m_loci)
# FST values for two subpops
inbr_subpops <- c(0.1, 0.3)
# matrix of intermediate subpop allele freqs
p_subpops <- draw_p_subpops(p_anc, inbr_subpops)

# special case of scalar p_anc
p_subpops <- draw_p_subpops(p_anc = 0.5, inbr_subpops, m_loci = m_loci)
stopifnot ( nrow( p_subpops ) == m_loci )

# special case of scalar inbr_subpops
k_subpops <- 2
p_subpops <- draw_p_subpops(p_anc, inbr_subpops = 0.2, k_subpops = k_subpops)
stopifnot ( ncol( p_subpops ) == k_subpops )

# both main parameters scalars but return value still matrix
p_subpops <- draw_p_subpops(p_anc = 0.5, inbr_subpops = 0.2, m_loci = m_loci, k_subpops = k_subpops)
stopifnot ( nrow( p_subpops ) == m_loci )
stopifnot ( ncol( p_subpops ) == k_subpops )

# passing scalar parameters without setting dimensions separately results in a 1x1 matrix
p_subpops <- draw_p_subpops(p_anc = 0.5, inbr_subpops = 0.2)
stopifnot ( nrow( p_subpops ) == 1 )
stopifnot ( ncol( p_subpops ) == 1 )
```

`fixed_loci` *Identify fixed loci*

Description

A locus is said to be fixed if the non-missing sub-vector contains all 0's or all 2's (the locus is completely homozygous for one allele or completely homozygous for the other allele). This function tests each locus, returning a vector that is TRUE for each fixed locus, FALSE otherwise. A locus with only missing elements (NA) will also be marked as fixed (TRUE). Below m is the number of loci, and n is the number of individuals.

Usage

```
fixed_loci(X)
```

Arguments

`X`	The $m \times n$ genotype matrix

Value

A length-m boolean vector where the i element is TRUE if locus i is fixed or completely missing, FALSE otherwise.

Examples

```
# here's a toy genotype matrix
X <- matrix(
       data = c(
              2, 2, NA, # fixed locus (with one missing element)
              0, NA, 0, # another fixed locus, for opposite allele
              1, 1, 1, # NOT fixed (heterozygotes are not considered fixed)
              0, 1, 2, # a completely variable locus
              NA, NA, NA # completely missing locus (will be treated as fixed)
             ),
       ncol = 3, byrow = TRUE)

# test that we get the desired values
stopifnot(
  fixed_loci(X) == c(TRUE, TRUE, FALSE, FALSE, TRUE)
)
```

`fst_admix` *Calculate FST for the admixed individuals*

Description

This function returns the F_{ST} of the admixed individuals given the admixture proportion matrix for n individuals and k intermediate subpopulations, the coancestry matrix of intermediate subpopulations (or its special cases, see `coanc_subpops` parameter below), and optional weights for individuals. This F_{ST} equals the weighted mean of the diagonal of the coancestry matrix (see `coanc_admix`).

Usage

```
fst_admix(admix_proportions, coanc_subpops, weights = NULL)
```

Arguments

`admix_proportions`	The $n \times k$ admixture proportion matrix
`coanc_subpops`	Either the $k \times k$ intermediate subpopulation coancestry matrix (for the complete admixture model), or the length-k vector of intermediate subpopulation F_{ST} values (for the BN-PSD model), or a scalar F_{ST} value shared by all intermediate subpopulations.
`weights`	The length-n vector of weights for individuals that define F_{ST} (default uniform weights)

Value

The F_{ST} of the admixed individuals

Examples

```
# set desired parameters
# number of individuals
n_ind <- 1000
# number of intermediate subpopulations
k_subpops <- 10

# differentiation of intermediate subpopulations
coanc_subpops <- ( 1 : k_subpops ) / k_subpops

# construct admixture proportions
admix_proportions <- admix_prop_1d_linear(n_ind, k_subpops, sigma = 1)

# lastly, calculate Fst!!! (uniform weights in this case)
fst_admix(admix_proportions, coanc_subpops)
```

`make_p_ind_admix` *Construct individual-specific allele frequency matrix under the PSD admixture model*

Description

Here m is the number of loci, n the number of individuals, and k the number of intermediate subpopulations. The $m \times n$ individual-specific allele frequency matrix P is constructed from the $m \times k$ intermediate subpopulation allele frequency matrix B and the $n \times k$ admixture proportion matrix Q using

$$P = BQ^T.$$

This function is a wrapper around `tcrossprod`, but also ensures the output allele frequencies are in [0, 1], as this is not guaranteed by `tcrossprod` due to limited machine precision.

Usage

```
make_p_ind_admix(p_subpops, admix_proportions)
```

Arguments

`p_subpops`	The $m \times k$ matrix of intermediate subpopulation allele frequencies.
`admix_proportions`	The $n \times k$ matrix of admixture proportions.

Value

The $m \times n$ matrix of individual-specific allele frequencies.

Examples

```
# data dimensions
# number of loci
m_loci <- 10
# number of individuals
n_ind <- 5
# number of intermediate subpops
k_subpops <- 2

# FST values for k = 2 subpops
inbr_subpops <- c(0.1, 0.3)

# non-trivial admixture proportions
admix_proportions <- admix_prop_1d_linear(n_ind, k_subpops, sigma = 1)

# random vector of ancestral allele frequencies
p_anc <- draw_p_anc(m_loci)

# matrix of intermediate subpop allele freqs
```

```
p_subpops <- draw_p_subpops(p_anc, inbr_subpops)

# matrix of individual-specific allele frequencies
p_ind <- make_p_ind_admix(p_subpops, admix_proportions)
```

Index