# Package 'bnpsd'

February 12, 2021

**Title** Simulate Genotypes from the BN-PSD Admixture Model

**Version** 1.2.3

**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 (2021) <doi:10.1371/journal.pgen.1009241>.

**Depends**

**Imports** stats

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

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.1

**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** 2021-02-12 09:50:12 UTC

# R topics documented:



---

`admix_prop_1d_circular` *Construct admixture proportion matrix for circular 1D geography*

---

## Description

Assumes `k_subpops` 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_ind` individuals sampled equally spaced in `[coord_ind_first,coord_ind_last]` (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 is `sigma = 1 / sqrt(kappa)` of the Von Mises density. If `sigma` is missing, it can be set indirectly by providing three variables: (1) the desired bias coefficient `bias_coeff`, (2) the coancestry matrix of the intermediate subpopulations `coanc_subpops` (up to a scalar factor), and (3) the final `fst` 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 = NA,
  coanc_subpops = NULL,
  fst = NA
)
```

## 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 | If sigma is NA, this bias coefficient is required. |
| coanc_subpops | If sigma is NA, this intermediate subpops coancestry is required. It can be provided as a k_subpops-by-k_subpops matrix, a length-k_subpops population inbreeding vector (for independent subpopulations, where between-subpop coancestries are zero) or scalar (if population inbreeding values are all equal and coancestries are zero). This coanc_subpops can be in the wrong scale (it cancels out in calculations), which is returned corrected, to result in the desired fst (next). |
| fst | If sigma is NA, this FST of the admixed individuals is required. |

## 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 (\link{admix_prop_1d_linear}).

If sigma is NA, its value is determined from the desired bias_coeff, coanc_subpops up to a scalar factor, and fst. Uniform weights for the final generalized FST 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 FST. However, the function stops if any rescaled coanc_subpops values are greater than 1, which are not allowed since they are IBD probabilities.

## Value

If sigma was provided, the n_ind-by-k_subpops admixture proportion matrix (admix_proportions). If sigma is missing, a named list is returned containing admix_proportions, the rescaled coanc_subpops, and the sigma (which together give the desired bias_coeff and fst).

## Examples

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

```
# advanced version: a similar model but with a bias coefficient of exactly 1/2
# (must provide bias_coeff, coanc_subpops, and fst in lieu of sigma)
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_subpops` intermediate subpopulations placed along a line at locations `1 : k_subpops` spread by random walks, then `n_ind` individuals equally spaced in [`coord_ind_first`,`coord_ind_last`] 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 standard deviation of the Normal densities) is `sigma`. If `sigma` is missing, it can be set indirectly by providing three variables: (1) the desired bias coefficient `bias_coeff`, (2) the coancestry matrix of the intermediate subpopulations `coanc_subpops` (up to a scalar factor), and (3) the final `fst` 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 = NA,
  coanc_subpops = NULL,
  fst = NA
)
```

## 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 (default 0.5). |
| coord_ind_last | Location of last individual (default k_subpops + 0.5). |
| | OPTIONS FOR BIAS COEFFICIENT VERSION |
| bias_coeff | If sigma is NA, this bias coefficient is required. |
| coanc_subpops | If sigma is NA, this intermediate subpops coancestry is required. It can be provided as a k_subpops-by-k_subpops matrix, a length-k_subpops population inbreeding vector (for independent subpopulations, where between-subpop coancestries are zero) or scalar (if population inbreeding values are all equal and coancestries are zero). This coanc_subpops can be in the wrong scale (it cancels out in calculations), which is returned corrected, to result in the desired fst (next). |
| fst | If sigma is NA, this FST of the admixed individuals is required. |

## Details

If sigma is NA, its value is determined from the desired bias_coeff, coanc_subpops up to a scalar factor, and fst. Uniform weights for the final generalized FST 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 FST. However, the function stops if any rescaled coanc_subpops values are greater than 1, which are not allowed since they are IBD probabilities.

## Value

If sigma was provided, the n_ind-by-k_subpops admixture proportion matrix (admix_proportions). If sigma is missing, a named list is returned containing admix_proportions, the rescaled coanc_subpops, and the sigma (which together give the desired bias_coeff and fst).

## Examples

```
# admixture matrix for 1000 individuals drawing alleles from 10 subpops
# simple version: spread of 2 standard deviations along the 1D geography
# (just set sigma)
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)

# advanced version: a similar model but with a bias coefficient of exactly 1/2
# (must provide bias_coeff, coanc_subpops, and fst in lieu of sigma)
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 = sort(unique(labs)))
```

## 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`-by-`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 FST estimation. Simulated loci are drawn independently (in linkage equilibrium).

## Author(s)

**Maintainer**: Alejandro Ochoa <alejandro.ochoa@duke.edu> (ORCID)

Authors:

- John D. Storey <jstorey@princeton.edu> (ORCID)

## 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

The n-by-n coancestry matrix `Theta` of admixed individuals is determined by the n-by-k admixture proportion matrix `Q` and the k-by-k intermediate subpopulation coancestry matrix `Psi`, given by `Theta = Q %*% Psi %*% t(Q)` In the more restricted BN-PSD model, `Psi` is a diagonal matrix (with FST values for the intermediate subpopulations along the diagonal, zero values off-diagonal).

## Usage

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

### Arguments

admix_proportions
: The n-by-k admixture proportion matrix

coanc_subpops
: The intermediate subpopulation coancestry, given either as a k-by-k matrix (for the complete admixture model), or the length-k vector of intermediate subpopulation FST values (for the BN-PSD model; implies zero coancestry between subpopulations), or a scalar FST value shared by all intermediate subpopulations (also implies zero coancestry between subpopulations).

### Value

The n-by-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

If `Theta` is the coancestry matrix and `Phi` is the kinship matrix (both are n-by-n symmetric), then these matrices agree off-diagonal, but the diagonal gets transformed as `diag( Phi ) = ( 1 + diag( Theta ) ) / 2`.

## Usage

```
coanc_to_kinship(coancestry)
```

## Arguments

| | |
|---|---|
| coancestry | The n-by-n coancestry matrix |

## Value

The n-by-n kinship matrix, preserving column and row names.

## See Also

The inverse function is given by \link[popkin]{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 FST values. The function is a wrapper around \link{draw_p_anc}, \link{draw_p_subpops}, \link{make_p_ind_admix}, and \link{draw_genotypes_admix} with additional features such as requiring polymorphic loci. Importantly, by default fixed loci (where all individuals were homozygous for the same allele) 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_loci` (also `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,
  verbose = FALSE,
  require_polymorphic_loci = TRUE,
  beta = NA,
  p_anc = NULL
)
```

## Arguments

| | |
|---|---|
| `admix_proportions` | The n-by-k matrix of admixture proportions. |
| `inbr_subpops` | The length-k vector (or scalar) of intermediate subpopulation FST 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. Note that by default `p_ind` is not constructed in full at all, instead a fast low-memory algorithm constructs it in parts as needed only; beware that setting `want_p_ind = TRUE` increases memory usage in comparison. |
| `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 vector of random ancestral allele frequencies in the return list. |
| `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_loci`).

`beta` Shape parameter for a symmetric Beta for ancestral allele frequencies `p_anc`. If `NA` (default), `p_anc` is uniform with range in [0.01, 0.5]. Otherwise, `p_anc` has a symmetric Beta distribution with range in [0, 1].

`p_anc` If provided, it is used as the ancestral allele frequencies (instead of drawing random ones). Must either be a scalar or a length-`m_loci` vector. If scalar and `want_p_anc = TRUE`, then the returned `p_anc` is the scalar value repeated `m_loci` times (it is always a vector).

## Value

A named list with the following items (which may be missing depending on options):

- `X`: An `m`-by-`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`-by-`k` matrix of intermediate subpopulation allele frequencies Included if `want_p_subpops = TRUE`.
- `p_ind`: An `m`-by-`n` matrix of individual-specific allele frequencies. Included if `want_p_ind = TRUE`.

## 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$p_subpops
#
# # ... `want_p_ind == TRUE`
# # individual-specific AFs
# p_ind <- out$p_ind
```

---

`draw_genotypes_admix` *Draw genotypes from the admixture model*

---

## Description

Given the Individual-specific Allele Frequency (IAF) matrix `p_ind` for `m` loci (rows) and `n` individuals (columns), the genotype matrix `X` (same dimensions as `p_ind`) is drawn from the Binomial distribution equivalent to `X[ i,j ] <-rbinom( 1,2,p_ind[ i,j ] )`, except the function is more efficient. If `admix_proportions` is provided as the second argument (a matrix with `n` individuals along rows and `k` intermediate subpopulations along the columns), the first argument `p_ind` is treated as the intermediate subpopulation allele frequency matrix (must be `m`-by-`k`) and the IAF matrix is equivalent to `p_ind %*% t( admix_proportions )`. However, in this case the function computes the IAF matrix in parts only, never stored in full, greatly reducing memory usage. If `admix_proportions` is missing, then `p_ind` is treated as the IAF matrix.

## Usage

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

## Arguments

| | |
|---|---|
| `p_ind` | The `m`-by-`n` IAF matrix (if `admix_proportions` is missing) or the `m`-by-`k` intermediate subpopulation allele frequency matrix (if `admix_proportions` is present) |
| `admix_proportions` | The optional `n`-by-`k` admixture proportion matrix (to draw data from the admixture model using reduced memory, by not fully forming the IAF matrix) |

## Value

The `m`-by-`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 computed internally in parts, never stored in full,
# reducing memory use substantially)
X_ind <- draw_genotypes_admix(p_subpops, admix_proportions)
```

---

`draw_p_anc` *Draw random Uniform or Beta ancestral allele frequencies*

---

## Description

This is simply a wrapper around \link[stats]{runif} or \link[stats]{rbeta} (depending on parameters) with different defaults and additional validations.

## Usage

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

## Arguments

| | |
|---|---|
| `m_loci` | Number of loci to draw. |
| `p_min` | Minimum allele frequency to draw (Uniform case only). |
| `p_max` | Maximum allele frequency to draw (Uniform case only). |
| `beta` | Shape parameter for a symmetric Beta. If `NA` (default), Uniform(`p_min`, `p_max`) is used. Otherwise, a Symmetric Beta is used and the user-specified range is ignored (values in [0, 1] will be returned). |

## Value

A length-m vector of random ancestral allele frequencies

## Examples

```
# Default is uniform with range between 0.01 and 0.5
p_anc <- draw_p_anc(m_loci = 10)

# Use of `beta` triggers a symmetric Beta distribution.
# This parameter has increased density for rare minor allele frequencies,
# resembling the 1000 Genomes allele frequency distribution
p_anc <- draw_p_anc(m_loci = 10, beta = 0.03)
```

---

# draw_p_subpops *Draw allele frequencies for independent subpopulations*

---

## Description

The allele frequency matrix P for m_loci loci (rows) and k_subpops independent subpopulations (columns) are drawn from the Balding-Nichols distribution with ancestral allele frequencies p_anc and FST parameters inbr_subpops equivalent to P[ i,j ] <-rbeta( 1,nu_j * p_anc[i],nu_j * ( 1 -p_anc[i] ) ), where nu_j <-1 / inbr_subpops[j] -1. The actual function is more efficient than the above code.

## Usage

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

## Arguments

| | |
|---|---|
| p_anc | The scalar or length-m_loci vector of ancestral allele frequencies per locus. |
| inbr_subpops | The length-k_subpops vector of subpopulation FST values. |
| m_loci | If p_anc is scalar, optionally provide the desired number of loci (lest only one locus be simulated). Stops if both length(p_anc) > 1 and m_loci is not NA and they disagree. |
| k_subpops | If inbr_subpops is a scalar, optionally provide the desired number of subpopulations (lest a single subpopulation be simulated). Stops if both length(inbr_subpops) > 1 and k_subpops is not NA and they disagree. |

## Value

The m_loci-by-k_subpops 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 "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. Loci with only missing elements (NA) are treated as fixed. Below m is the number of loci, and n is the number of individuals.

## Usage

```
fixed_loci(X)
```

## Arguments

| | |
|---|---|
| X | The m-by-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 generalized FST of the admixed individuals given their admixture proportion matrix, the coancestry matrix of intermediate subpopulations (or its special cases, see coanc_subpops parameter below), and optional weights for individuals. This FST equals the weighted mean of the diagonal of the coancestry matrix (see \link{coanc_admix}). Below there are n individuals and k intermediate subpopulations.

## Usage

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

## Arguments

| | |
|---|---|
| admix_proportions | The n-by-k admixture proportion matrix |
| coanc_subpops | Either the k-by-k intermediate subpopulation coancestry matrix (for the complete admixture model), or the length-k vector of intermediate subpopulation FST values (for the BN-PSD model; assumes zero coancestries between subpopulations), or a scalar FST value shared by all intermediate subpopulations (also assumes zero coancestry between subpopulations). |
| weights | Optional length-n vector of weights for individuals that define their generalized FST (default uniform weights) |

## Value

The generalized FST 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`-by-`n` individual-specific allele frequency matrix `p_ind` is constructed from the `m`-by-`k` intermediate subpopulation allele frequency matrix `p_subpops` and the `n`-by-`k` admixture proportion matrix `admix_proportions` equivalent to `p_ind <-p_subpops %*% t( admix_proportions )`. This function is a wrapper around \link{tcrossprod}, but also ensures the output allele frequencies are in [0, 1] (otherwise not guaranteed due to limited machine precision).

## Usage

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

## Arguments

| | |
|---|---|
| `p_subpops` | The `m`-by-`k` matrix of intermediate subpopulation allele frequencies. |
| `admix_proportions` | The `n`-by-`k` matrix of admixture proportions. |

## Value

The `m`-by-`n` matrix of individual-specific allele frequencies `p_ind`.

## 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