Package: AdmixPoly (via r-universe)

June 9, 2026

Type Package

Title Global and Local Admixture Inference in Polyploids

Version 1.0.0

Description Provides functions to perform global (genome-wide) and local admixture inference from bi- and multi-allelic marker dosages (discrete or continuous) in polyploid species.

License GPL (>= 3)

Encoding UTF-8

NeedsCompilation yes

Imports Rcpp, purrr, utils, dplyr (>= 1.1.0), tidyr, tibble, ggplot2, magrittr, Matrix, data.table

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.3.2

Depends R (>= 3.6.0)

Suggests knitr, rmarkdown

VignetteBuilder knitr

Language en-US

SystemRequirements A system with a C++ compiler supporting OpenMP

Author Simon Rio [aut, cre] (ORCID: <https://orcid.org/0000-0001-7014-8789>), Tristan Mary-Huard [aut] (ORCID: <https://orcid.org/0000-0002-3839-9067>), Franck Gauthier [aut] (ORCID: <https://orcid.org/0000-0003-0574-065X>)

Maintainer Simon Rio <simon.rio@cirad.fr>

Config/pak/sysreqs libicu-dev

Repository https://simonrio.r-universe.dev

Date/Publication 2026-06-08 19:30:20 UTC

RemoteUrl https://github.com/cran/AdmixPoly

RemoteRef HEAD

RemoteSha bf647eb5dd7c24f15c217ca3c6ca377a384ba412

Contents

`AdmixPoly-package` *AdmixPoly: Global and Local Admixture Inference in Polyploids*

Description

Provides functions to perform global and local admixture inference from bi- and multi-allelic marker dosages (discrete or continuous) in polyploid species.

Author(s)

Maintainer: Simon Rio <simon.rio@cirad.fr> (ORCID)

Authors:

- Tristan Mary-Huard <tristan.mary-huard@agroparistech.fr> (ORCID)
- Franck Gauthier <franck.gauthier@inrae.fr> (ORCID)

`AdmixGlobal` *Expectation-Maximization algorithm for global (genome-wide) admixture inference*

Description

This function performs global admixture inference from discrete or continuous allele dosages

Usage

```
AdmixGlobal(
  Geno,
  K,
  P = NULL,
  ParamToUpdate = "both",
  MaxIter = 200L,
  MinIter = 10L,
  MinParamBound = 1e-06,
  LogLikThresh = 0.001,
  PropInit = NULL,
  FreqInit = NULL,
  NbThreads = 0L,
  Seed = 123L,
  Verbose = TRUE
)
```

Arguments

Geno	List of genotying matrices of size M (number of markers), with each matrix of dimension N (number of individuals) x L (number of alleles) whose elements consists of discrete or continuous allele dosages
K	Number of ancestral groups (positive integer superior or equal to 2)
P	Ploidy level (positive integer), to be used when read depth ratios are specified in Geno, either as a single value to specify the same ploidy for all individuals, or as a vector of size N
ParamToUpdate	Specify both, Prop or Freq to update both admixture proportions and ancestral allele frequencies, only admixture proportions or only ancestral allele frequencies, respectively
MaxIter	Maximum number of iterations (positive integer greater than or equal to MinIter)
MinIter	Minimum number of iterations (positive integer greater than or equal to 2 and smaller than or equal to MaxIter)
MinParamBound	Minimum value for admixture proportions and ancestral allele frequencies (positive numeric value)
LogLikThresh	Algorithm convergence criterion (positive numeric value) consisting of a log-likelihood difference value between two iterations
PropInit	Matrix of dimension N x K with initial admixture proportions
FreqInit	List of matrices of size M (number of markers), with each matrix being of dimension K x L with initial allele frequencies in ancestral groups
NbThreads	Number of threads to be used (positive integer) with a default value of 0 setting automatically all threads available
Seed	Seed for reproducible inference (integer)
Verbose	A boolean describing if detailed information should be printed

Details

The function `AdmixGlobal()` performs global (genome-wide) admixture inference from genotyping data (`Geno`) formatted as a list of matrices (one for each marker), by specifying the number of ancestral groups (`K`).

The inference is performed using an expectation-maximization (EM) algorithm whose minimum (`MinIter`) and maximum (`Maxiter`) number of iterations can be fixed by the user, as well as the log-likelihood convergence threshold (`LogLikThresh`). A minimum value for admixture proportions ancestral allele frequencies is set using `MinParamBound`, which should be above zero to prevent computational issues. By default, all available threads/CPU cores are used but the number can be chosen using `NbThreads`.

If allele dosages are used, the ploidy level (`P`) does not have to be specified as it is automatically calculated from the dosages. However, it must be specified if the genotypic data correspond to ratios constrained between 0 and 1 (e.g. obtained from allele read depths).

By default, the EM algorithm is initialized with random parameter values but admixture proportions and ancestral allele frequencies can be initialized using `PropInit` and `FreqInit`, respectively.

By default, both types of parameters are updated by the EM, but only one of them can be updated while the other is fixed using `ParamToUpdate`. This is particularly interesting when allele frequencies have been estimated on a reference panel (e.g. stored into a list `FreqRef`) and are used to estimated the admixture proportions of a new panel of individuals. This scenario can be implemented by initializing allele frequencies (`FreqInit=FreqRef`) and specifying to update only admixture proportions (`ParamToUpdate=Prop`).

Value

A list of three items: a matrix of admixture proportions (`Prop`), a list of matrices of allele frequencies in ancestral groups (`Freq`), and a vector of log-likelihood values over iterations (`LogLik`)

See Also

- `SimulatePop()` to simulate a polyploid admixed population.
- `AdmixLocal()` to perform local admixture inference using the results from the `AdmixGlobal()` function.
- `GlobalPlot()` to generate an admixture barplot using the results from the `AdmixGlobal()` function.

Examples

```
## Simulate a polyploid admixed population
DataSim <- SimulatePop(K=3L, N=10L, P=6L, M=50L, C=5L, L=10L, Seed=123, NbThreads=1)

## Perform global admixture inference
ResGlobalAdmix <- AdmixGlobal(Geno=DataSim$Geno, K=3, Verbose=FALSE, NbThreads=1)

## Estimated admixture proportions
head(ResGlobalAdmix$Prop)

## Estimated allele frequencies at first marker
ResGlobalAdmix$Freq[[1]]
```

```
## Log-likelihood over iterations
head(ResGlobalAdmix$LogLik)
```

`AdmixLocal` *Forward-Backward algorithm for local admixture inference*

Description

This function performs local admixture inference for a selected individual

Usage

```
AdmixLocal(
  Geno,
  ResAdmixGlobal,
  Ind,
  P,
  GeneticMap,
  SmoothParam = 1,
  MaxRec = NULL,
  DistIntBounds = c(0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1),
  MinProp = 1e-06,
  EmisProbMaxPermut = 100L,
  NbThreads = 0L,
  Verbose = TRUE
)
```

Arguments

`Geno`	List of genotying matrices of size `M` (number of markers), with each matrix of dimension `N`(number of individuals) x `L`(number of alleles) whose elements consists of discrete or continuous dosages
`ResAdmixGlobal`	A list generated by the `AdmixGlobal` function
`Ind`	Name of the individual to be processed
`P`	Ploidy level of the individual (positive integer superior or equal to 2)
`GeneticMap`	Dataframe with three columns: `Chromosome`, `Marker`, and `Distance`, with `M` rows. Distances must be in centiMorgan
`SmoothParam`	Smoothing parameter corresponding to the number of generations since admixture
`MaxRec`	Maximum number of recombination between adjacent markers, used as an approximation for transition probabilities, in order to speed up calculations and limit memory usage. The approximation is disabled by default. Specify a positive integer between 1 and `P` to activate it.

DistIntBounds	Vector of bounds for genetic distance intervals, used as an approximation for transition probabilities, in order to speed up calculations and limit memory usage. To deactivate the approximation, set DistIntBounds=NULL.
MinProp	Minimum value for admixture proportion below which the contribution of the group is not considered for the individual (positive numeric value inferior to 1/K where K is the number of groups)
EmisProbMaxPermut	Maximum number of permutations involved in computing exact emission probability, otherwise an approximate emission probability is calculated, used as an approximation to speed up calculations. When continuous dosages are informed, the approximate emission probability is calculated.
NbThreads	Number of threads to be used (positive integer) with a default value of 0 setting automatically all threads available
Verbose	A boolean describing if detailed information should be printed

Details

The function AdmixLocal() performs local admixture inference from: (i) genotyping data (Geno) formatted as a list of matrices (one for each marker), (ii) a list (ResAdmixGlobal) generated by the global admixture function AdmixGlobal(), (iii) a genetic map dataframe (GeneticMap), (iV) the name of the individual to analyse (Ind), (v) the ploidy level of the individual (P).

The inference of the local admixture hidden Markov model is performed by calculating posterior probabilities and the Viterbi path, both available from the results. By default, all available threads/CPU cores are used but the number can be chosen using NbThreads.

The rate at which breakpoints between blocks of homogeneous ancestry occur per unit of genetic distance is defined as a smoothing parameter controlling the size of ancestry blocks (SmoothParam). The larger the value, the smaller the expected ancestry block size.

Two approximations exist for transition probabilities to speedup calculations: (i) the first consists of a simplification of genetic distances between adjacent marker where genetic distances (in cM) comprised within each of these intervals are averaged (DistIntBounds), (ii) the second consists of limiting the number of recombination authorized to move from one ancestry state to another between adjacent markers (MaxRec). When both arguments are specified as NULL, exact transition probabilities are calculated.

When allele dosages are discrete, the exact emission probability distribution function is calculated when the number of phased genotypes (i.e. permutations) that can be derived from the unphased allele dosage does not exceed a threshold defined by EmisProbMaxPermut. When number of permutations exceeds the threshold or when allele dosages are continuous (e.g. obtained from read depth ratios), an approximation of the emission probability distribution is calculated, which helps speeding up calculations.

When an individual has a very low global (genome-wide) admixture proportion for an ancestral group, this contribution can be considered negligible and discarded from the local admixture inference to speedup calculations. The minimum threshold can be set using MinProp.

Value

A list of four items: a list of size C (number of chromosomes) with ancestry dosage matrices obtained from the posterior distribution (Posterior), a list of size C with ancestry dosages obtained

from the Viterbi algorithm (`Viterbi`), the log-likelihood value (`LogLik`), and a list of size `C` of vectors indicating what emission probability was computed at each marker (`EmisProbType`)

See Also

- `SimulatePop()` to simulate a polyploid admixed population.
- `AdmixGlobal()` to perform global (genome-wide) admixture inference and generate the `ResAdmixGlobal` object for the `AdmixLocal()` function.
- `LocalPlot()` to generate a local admixture plot using the results from the `AdmixLocal()` function.

Examples

```
## Simulate Simulate a polyploid admixed population
DataSim <- SimulatePop(K=3L, N=10L, P=6L, M=50L, C=5L, L=10L, Seed=123, NbThreads=1)

## Perform global admixture inference
ResAdmixGlobal <- AdmixGlobal(Geno=DataSim$Geno, K=3, Verbose=FALSE, NbThreads=1)

## Perform local admixture inference for one individual
ResAdmixLocal <- AdmixLocal(Geno=DataSim$Geno, ResAdmixGlobal, "Ind4", 6L,
                            DataSim$GeneticMap, Verbose=FALSE, NbThreads=1)

## Posterior ancestry dosages for Chr1
head(ResAdmixLocal$Posterior$Chr1)

## Viterbi ancestry dosages for Chr1
head(ResAdmixLocal$Viterbi$Chr1)

## Log-likelihood
ResAdmixLocal$LogLik

## Type of emission probability for Chr1
head(ResAdmixLocal$EmisProbType$Chr1)
```

`ComputeLogLik` *Calculate the log-likelihood*

Description

Calculate the log-likelihood

Usage

```
ComputeLogLik(Geno, Prop, Freq, NbThreads = 0L)
```

Arguments

Geno	List of genotying matrices of size "number of markers", with each matrix of dimension "number of individuals" x "number of alleles" whose elements consists of discrete or continuous dosages
Prop	Matrix of admixture proportions of dimension "number of individuals" x "number of groups"
Freq	List of allele frequency matrices of size "number of markers", with each matrix of dimension "number of groups" x "number of alleles"
NbThreads	Number of threads to be used (positive integer) with 0 default values setting automatically all threads available

Value

log-likelihood

GlobalPlot *Plot global admixture proportions*

Description

This function generates an admixture barplot based on `ggplot2::ggplot()`

Usage

```
GlobalPlot(Prop)
```

Arguments

Prop	Matrix of admixture proportions of dimension N (number of individuals) x K (number of groups)

Value

A global admixture barplot as a ggplot object

See Also

- `SimulatePop()` to simulate a polyploid admixed population.
- `AdmixGlobal()` to perform global (genome-wide) admixture inference and generate the Prop object.

Examples

```
## Simulate a polyploid admixed population
DataSim <- SimulatePop(K=3L, N=10L, P=6L, M=50L, C=5L, L=10L, Seed=123, NbThreads=1)

## Perform global admixture inference
ResGlobalAdmix <- AdmixGlobal(Geno=DataSim$Geno, K=3, Verbose=FALSE, NbThreads=1)

## Admxiture barplot
GlobalPlot(ResGlobalAdmix$Prop)
```

`LocalPlot` *Plot local ancestry dosages*

Description

This function generates local admixture plot

Usage

```
LocalPlot(Ancestry, GeneticMap, DisplayNAs = FALSE)
```

Arguments

`Ancestry`	List of local ancestry dosages for an individual of size `C` (number of chromosomes) with each element being a matrix with ancestry dosages of dimension `M` (number of markers) x `K` (number of groups)
`GeneticMap`	Dataframe with three columns: `Chromosome`, `Marker`, and `Distance`, with `M` rows. Distance must be in centiMorgan
`DisplayNAs`	Should missing data be displayed as gaps (FALSE by default)

Details

The function `LocalPlot()` based on `ggplot2::ggplot()` performs local admixture plot using local ancestry dosage estimated from the `AdmixLocal()` function and a genetic map.

The plot displays local ancestry dosages for one individual, with one facet for each of the chromosomes. The x-axis represents the genetic distance and the y-axis represents ancestry dosages, ranging from 0 to the ploidy level of the individual.

When the individual has missing genotypic data as compared to the genetic map, the gaps can be displayed as blanks by setting `DisplayNAs=TRUE`

Value

A local admixture plot as a ggplot object

See Also

- SimulatePop() to simulate a polyploid admixed population.
- AdmixGlobal() to perform global (genome-wide) admixture inference.
- AdmixLocal() to perform local admixture inference.

Examples

```
## Simulate Simulate a polyploid admixed population
DataSim <- SimulatePop(K=3L, N=10L, P=6L, M=50L, C=5L, L=10L, Seed=123, NbThreads=1)

## Perform global admixture inference
ResAdmixGlobal <- AdmixGlobal(Geno=DataSim$Geno, K=3, Verbose=FALSE, NbThreads=1)

## Perform local admixture inference for one individual
ResAdmixLocal <- AdmixLocal(Geno=DataSim$Geno, ResAdmixGlobal, "Ind4", 6L,
                            DataSim$GeneticMap, Verbose=FALSE, NbThreads=1)

## Local admixture barplot
LocalPlot(ResAdmixLocal$Posterior, DataSim$GeneticMap)
```

ReadHPA *Read Haplotype Presence-Absence (HPA) file*

Description

This function reads and format HPA files generated by HaploCharmer.

Usage

```
ReadHPA(
  File,
  MaxMarkerMissing = 0.2,
  MaxIndMissing = 0.2,
  TotalDepthField = "DP",
  AlleleDepthField = "AD",
  NbThreads = 0L,
  Verbose = TRUE
)
```

Arguments

File	Path to HPA file
MaxMarkerMissing	Maximum proportion of missing values for marker above which the marker is discarded
MaxIndMissing	Maximum proportion of missing values for individual above which the individual is discarded (applied after marker filtering)

`TotalDepthField`
: Total read depth FORMAT field in the HPA file to get allele depth proportions

`AlleleDepthField`
: Alternative allele read depth FORMAT field in the HPA file to get allele depth proportions

`NbThreads`
: Number of threads to be used (positive integer) with a default value of 0 setting automatically all threads available

`Verbose`
: A boolean describing if detailed information should be printed

Details

The function `ReadHPA()` reads Hapotype Presence-Absence (HPA) files generated by HaploCharmer.

This function generates genotypic data coded as haplotype read depth ratios using the read depths from the haplotype (`AlleleDepthField`) and total read depth (`TotalDepthField`) fields of the HPA file.

Filtering of missing data can be applied to individuals and markers by setting the maximum percentages `MaxMarkerMissing` and `MaxIndMissing`, respectively.

By default, all available threads/CPU cores are used but the number can be chosen using `NbThreads`.

Value

A list of three items: a filtered list of genotying matrices (`Geno`), a dataframe with variant information (`MarkerInfo`), and a dataframe to be used as a genetic map proxy for local admixture inference (`GeneticMap`). The genetic map proxy is based on physical positions assuming all chromosomes are 100cM long.

See Also

`ReadHPA()` to read VCF files

Examples

```
## Read HPA
hpa_path <- system.file("extdata", "Test.hpa", package = "AdmixPoly")
DataHPA <- AdmixPoly::ReadHPA(File = hpa_path, NbThreads=1)
```

ReadVCF *Read VCF file*

Description

This function reads and format VCF files

Usage

```
ReadVCF(
  File,
  AlleleDepthField = NULL,
  AlleleDepthType = "alleles",
  MaxMarkerMissing = 0.2,
  MaxIndMissing = 0.2,
  NbThreads = 0L,
  Verbose = TRUE
)
```

Arguments

`File`	Path to VCF file
`AlleleDepthField`	Allele read depth FORMAT field in the VCF to get allele depth proportions. If NULL (default), dosage is obtained from GT field
`AlleleDepthType`	Allele read depth type: either the depths associated with the REF and ALT alleles (alleles, by default), or the depths associated with each haplotypes reported in the GT field (haplotypes). The latter can be used for a phased VCF generated by HaploCharmer.
`MaxMarkerMissing`	Maximum proportion of missing values for marker above which the marker is discarded
`MaxIndMissing`	Maximum proportion of missing values for individual above which the individual is discarded (applied after marker filtering)
`NbThreads`	Number of threads to be used (positive integer) with a default value of 0 setting automatically all threads available
`Verbose`	A boolean describing if detailed information should be printed

Details

The function `ReadVCF()` reads Variant Call Format (VCF) files.

By default, the dosages are obtained from the GT field of the VCF. However, genotypic data are commonly "diploidized" in polyploid species where all heterozygous classes are aggregated into one class. In this case, it is recommended to specify the allele depth field of the VCF to work with allele read depth ratios instead, e.g. `AlleleDepthField=AD`.

If the VCF has been generated by the workflow HaploCharmer where alleles are phased within blocks, the read depths are not associated to alleles but to haplotypes. In this case, read depth ratios associated to each allele can be obtained by specifying `AlleleDepthType="haplotypes"`.

Filtering of missing data can be applied to individuals and markers by setting the maximum percentages `MaxMarkerMissing` and `MaxIndMissing`, respectively.

By default, all available threads/CPU cores are used but the number can be chosen using `NbThreads`.

Value

A list of three items: a filtered list of genotying matrices (`Geno`), a dataframe with variant information (`MarkerInfo`), and a dataframe to be used as a genetic map proxy for local admixture inference (`GeneticMap`). The genetic map proxy is based on physical positions assuming all chromosomes are 100cM long.

See Also

`ReadHPA()` to read Haplotype Presence-Absence files generated by HaploCharmer.

Examples

```
## Read test VCF
vcf_path <- system.file("extdata", "Test.vcf", package = "AdmixPoly")
DataVCF <- AdmixPoly::ReadVCF(File = vcf_path, NbThreads=1)
```

`SimulatePop` *Simulate an admixed polyploid population characterized for multi-allelic markers*

Description

This function simulates a polyploid admixed population

Usage

```
SimulatePop(
  K = 3L,
  N = 100L,
  P = 6L,
  M = 1000L,
  C = 5L,
  L = 10L,
  SmoothParam = 1,
  GeneticMap = NULL,
  Prop = NULL,
  Freq = NULL,
  AlphaProp = 1,
  AlphaFreq = 1,
  Depth = NULL,
  SeqError = 0.01,
  NbThreads = 0L,
  Seed = 123L
)
```

Arguments

K	Number of ancestral groups (positive integer superior or equal to 2)
N	Number of individuals (positive integer superior or equal to 2)
P	Ploidy level (positive integer) either as a single value to specify the same ploidy for all simulated individuals, or as a vector of size N
M	Number of markers (positive integer superior or equal to 2)
C	Number of chromosomes (positive integer). By default, markers are evenly distributed between chromosomes and evenly spaced
L	Number of alleles (positive integer superior or equal to 2) either as a single value to specify the same number of alleles for all simulated markers, or as a vector of size M
SmoothParam	Smoothing parameter corresponding to the number of generations since admixture
GeneticMap	An optional genetic map that overrides the number of markers M and chromosomes C, with three named columns: Marker, Chromosome, Distance. The distances must be in centiMorgan
Prop	Preconfigured matrix of admixture proportions that overrides the number of individuals N and ancestral groups K
Freq	Preconfigured list of matrices of allele frequencies in ancestral groups that overrides the number of markers M, alleles L and ancestral groups K
AlphaProp	Concentration parameter from the Dirichlet distribution (positive numeric value) used for sampling admixture proportions
AlphaFreq	Concentration parameter from the Dirichlet distribution (positive numeric value) used for sampling allele frequencies in ancestral groups
Depth	(optional) Total read depth (positive integer), either as a single value to specify the same depth for all datapoints, or as a matrix of dimension N x M
SeqError	(optional) Sequencing error (numeric value between 0 and 1) rate to be used when sampling sequencing reads
NbThreads	Number of threads to be used (positive integer) with a default value of 0 setting automatically all threads available
Seed	Seed for reproducible inference (integer)

Value

A list of four items: a list of genotying matrices (Geno), a matrix of admixture proportions (Prop), a list of matrices of allele frequencies in ancestral groups (Freq), and a vector of log-likelihood values over iterations (LogLik)

Examples

```
## Simulate a polyploid admixed population
DataSim <- SimulatePop(K=3L, N=10L, P=6L, M=50L, C=5L, L=10L, Seed=123)

## Genotyping matrices
```

```
head(DataSim$Geno$Marker1)

## Ancestry matrices by chromosome
head(DataSim$Ancestry$Ind1$Chr1)

## Admixture proportions
head(DataSim$Prop)

## Allele frequencies in ancestral groups
DataSim$Freq$Marker1

## Genetic map
head(DataSim$GeneticMap)
```

`UpdateParam` *Update Prop and Freq parameters for EM step*

Description

Update Prop and Freq parameters for EM step

Usage

```
UpdateParam(
  Geno,
  Prop,
  Freq,
  ParamToUpdate = "both",
  MinParamBound = 1e-06,
  NbThreads = 0L
)
```

Arguments

`Geno`	List of genotying matrices of size "number of markers", with each matrix of dimension "number of individuals" x "number of alleles" whose elements consists of discrete or continuous dosages
`Prop`	Matrix of admixture coefficients of dimension "number of individuaks" x "number of groups"
`Freq`	List of allele frequency matrices of size "number of markers", with each matrix of dimension "number of groups" x "number of alleles"
`ParamToUpdate`	Specify "both", "Prop" or "Freq" to update both admixture proportions and allele frequencies, only admixture proportions or only allele frequencies, respectively
`MinParamBound`	Minimum bound value for parameters
`NbThreads`	Number of threads to be used (positive integer) with 0 default values setting automatically all threads available

Value

list of updated parameters with two elements: Prop and Freq

Index