Package 'Summix'

April 16, 2025

Title Summix2: A suite of methods to estimate, adjust, and leverage substructure in genetic summary data

Version 2.14.0

Description This package contains the Summix2 method for estimating and adjusting for substructure in genetic summary allele frequency data. The function summix() estimates reference group proportions using a mixture model. The adjAF() function produces adjusted allele frequencies for an observed group with reference group proportions matching a target individual or sample. The summix_local() function estimates local ancestry mixture proportions and performs selection scans in genetic summary data.

License MIT + file LICENSE

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Suggests rmarkdown, markdown, knitr, testthat (>= 3.0.0)

biocViews StatisticalMethod, WholeGenome, Genetics

VignetteBuilder knitr

Encoding UTF-8

Depends R (>= 4.3)

Imports dplyr, nloptr, magrittr, methods, tibble, tidyselect, BEDASSLE, scales, visNetwork, randomcoloR

LazyData true

BugReports https://github.com/Bioconductor/Summix/issues

Config/testthat/edition 3

git_url https://git.bioconductor.org/packages/Summix

git_branch RELEASE_3_21

git_last_commit 9a2ca60

git_last_commit_date 2025-04-15

Repository Bioconductor 3.21

Date/Publication 2025-04-15

Author Audrey Hendricks [cre],
Price Adelle [aut],
Stoneman Haley [aut]

Maintainer Audrey Hendricks <audrey.hendricks@cuanschutz.edu>

Contents

`adjAF` *adjAF*

Description

Adjusts allele frequencies for heterogeneous populations in genetic data given proportion of reference groups

Usage

```
adjAF(
  data,
  reference,
  observed,
  pi.target,
  pi.observed,
  adj_method = "average",
  N_reference = NULL,
  N_observed = NULL,
  filter = TRUE
)
```

Arguments

data	dataframe of unadjusted allele frequency for observed group, K reference group allele frequencies for N SNPs
reference	character vector of the column names for K reference groups.
observed	character value for the column name of observed data group
pi.target	numeric vector of the mixture proportions for K reference groups in the target individual or group.
pi.observed	numeric vector of the mixture proportions for K reference groups in the observed group.
adj_method	user choice of method for the allele frequency adjustment: options "average" and "leave_one_out" are available. Defaults to "average".
N_reference	numeric vector of the sample sizes for each of the K reference groups.
N_observed	numeric value of the sample size of the observed group.
filter	sets adjusted allele frequencies equal to 1 if > 1, to 0 if > -.005 and < 0, and removes adjusted allele frequencies < -.005.

Value

pi: table of input reference groups, pi.observed, and pi.target

observed.data: name of the data column for the observed group from which adjusted allele frequency is estimated

Nsnps: number of SNPs for which adjusted AF is estimated

adjusted.AF: data frame of original data with an appended column of adjusted allele frequencies

effective.sample.size: The sample size of individuals effectively represented by the adjusted allele frequencies

Author(s)

Adelle Price, <adelle.price@cuanschutz.edu>

Hayley Wolff, <hayley.wolff@cuanschutz.edu>

Audrey Hendricks, <audrey.hendricks@cuanschutz.edu>

References

https://github.com/hendriau/Summix2

See Also

https://github.com/hendriau/Summix2 for further documentation.

Examples

```
data(ancestryData)
adjusted_data<-adjAF(data   = ancestryData,
    reference  = c("reference_AF_afr", "reference_AF_eur"),
    observed    = "gnomad_AF_afr",
    pi.target   = c(1, 0),
    pi.observed = c(.85, .15),
    adj_method = 'average',
    N_reference = c(704,741),
    N_observed = 20744,
    filter = TRUE)
adjusted_data$adjusted.AF[1:5,]
```

`adjAF_calc` *adjAF_calc*

Description

Helper function for calculating allele frequencies for heterogeneous populations in genetic data given proportion of reference groups

Usage

```
adjAF_calc(data, reference, observed, pi.target, pi.observed)
```

Arguments

`data`	dataframe of unadjusted allele frequency for observed group, K-1 reference group allele frequencies for N SNPs
`reference`	character vector of the column names for K-1 reference groups. The name of the last reference group is not included as that group is not used to estimate the adjusted allele frequencies.
`observed`	character value for the column name of observed data group
`pi.target`	numeric vector of the mixture proportions for K reference groups in the target sample or subject. The order must match the order of the reference columns with the last entry matching the missing reference group.
`pi.observed`	numeric vector of the mixture proportions for K reference groups for the observed group. The order must match the order of the reference columns with the last entry matching the missing reference group.

Value

pi: table of input reference groups, pi.observed, and pi.target

observed.data: name of the data column for the observed group from which adjusted allele frequency is estimated

Nsnps: number of SNPs for which adjusted AF is estimated

adjusted.AF: data frame of original data with an appended column of adjusted allele frequencies

---

`ancestryData` *ancestryData*

Description

Sample dataset containing reference and observed allele frequencies to be used for examples within the Summix package.

Usage

```
ancestryData
```

Format

A data frame with 1000 rows (representing individual SNPs) and 10 columns:

POS Position of SNP on given chromosome.

REF Reference allele

ALT Alternate allele

CHROM Chromosome

reference_AF_afr Allele frequency column of the African reference ancestry.

reference_AF_eas Allele frequency column of the East Asian reference ancestry.

reference_AF_eur Allele frequency column of the European reference ancestry.

reference_AF_iam Allele frequency column of the Indigenous American reference ancestry.

reference_AF_sas Allele frequency column of the South Asian reference ancestry.

gnomad_AF_afr Allele frequency column of the observed gnomAD v3.1.2 African/African American population.

Source

https://gnomad.broadinstitute.org/downloads#v3

`calc_effective_N` *calc_effective_N*

Description

Helper function to calculate effective sample size for the group that is left out when estimating the adjusted allele frequencies in each adjAF function iteration.

Usage

```
calc_effective_N(N_reference, N_observed, pi.target, pi.observed)
```

Arguments

`N_reference`	numeric vector of the sample sizes of each K reference groups.
`N_observed`	numeric value of the sample size of the observed group.
`pi.target`	numeric vector of the mixture proportions for K reference groups in the target sample or subject. The order must match the order of the reference columns with the last entry matching the missing reference group.
`pi.observed`	numeric vector of the mixture proportions for K reference groups for the observed group. The order must match the order of the reference columns with the last entry matching the missing reference group.

Value

N_effective: effective sample size for the group that is left out when estimating the adjusted allele frequencies in each adjAF function iteration.

`calc_scaledObj` *calc_scaledObj*

Description

Helper function to calculate new scaled loss function using weighted AF bin objectives

Usage

```
calc_scaledObj(data, reference, observed, pi.start)
```

Arguments

data	a dataframe of the observed and reference allele frequencies for N genetic variants. See data formatting document at https://github.com/hendriau/Summix for more information. Uses the same input data as summix.
reference	a character vector of the column names for the reference groups.
observed	a string that is the column name for the observed group.
pi.start	Length K numeric vector of the starting guess for the reference group proportions. If not specified, this defaults to 1/K where K is the number of reference groups.

Value

numeric value that is the scaled objective per 1000 SNPs

---

doInternalSimulation *doInternalSimulation*

Description

Helper function to get the within block se using re-simulation

Usage

```
doInternalSimulation(windows, data, reference, observed, nRefs, nSim = 1000)
```

Arguments

windows	is a dataframe with the Start_Pos and End_Pos
data	is the original chromosome data
reference	is a list with the names of the columns with references
observed	a character value that is the column name for the observed group
nRefs	is a vector the same lengths as reference with the number of individuals in each reference population
nSim	is the number of internal simulations for the standard error calculations

getNextEndPoint *getNextEndPoint*

Description

Helper function: algorithm to get next end point in basic window algorithm; will find first point that is at least window size away from start

Usage

```
getNextEndPoint(data, start, windowSize)
```

Arguments

data	the input dataframe subset to the chromosome
start	index of the current start point
windowSize	the window size (in bp or variants)

Value

index of end point of window

getNextStartPoint *getNextStartPoint*

Description

Helper function: algorithm to get next start point; will pick the point that provides approx. the specified amount of overlap, but not more; if there are only two variants in the previous block, will jump new start point to the previous end point

Usage

```
getNextStartPoint(data, start, end, overlap)
```

Arguments

data	the input dataframe subset to the chromosome
start	the current index of start point
end	the current index of end point
overlap	the desired amount of window overlap (in bp or variants)

Value

returns index of new start point

saveBlock *saveBlock*

Description

Helper function to save one block to results

Usage

```
saveBlock(data, start, end, props, results)
```

Arguments

data	the input dataframe subsetting to just the chromosome
start	index of start of block
end	index of the end of block
props	substructure proportions for the block returned from summix
results	current results dataframe

sizeGetNext *sizeGetNext*

Description

Helper function to get starting end point that is a minimum distance (in bases) from start point; uses indices NOT position numbers

Usage

```
sizeGetNext(positions, start, minSize)
```

Arguments

positions	list of positions of variants
start	index of the current start position
minSize	integer defining the minimum size in bp of the window

Value

the new end point index

summix *summix*

Description

Estimating mixture proportions of reference groups from large (N SNPs>10,000) genetic AF data.

Usage

```
summix(
  data,
  reference,
  observed,
  pi.start = NA,
  goodness.of.fit = TRUE,
  override_removeSmallRef = FALSE,
  network = FALSE,
  N_reference = NA,
  reference_colors = NA
)
```

Arguments

data	A dataframe of the observed and reference allele frequencies for N genetic variants. See data formatting document at https://github.com/hendriau/Summix for more information.
reference	A character vector of the column names for the reference groups.
observed	A character value that is the column name for the observed group.
pi.start	Length K numeric vector of the starting guess for the reference group proportions. If not specified, this defaults to 1/K where K is the number of reference groups.
goodness.of.fit	Default value is TRUE. If set as FALSE, the user will override the default goodness of fit measure and return the raw objective loss from slsqp.
override_removeSmallRef	Default value is FALSE. If set as TRUE, the user will override the automatic removal of reference groups with <1% global proportions - this is not recommended.
network	Default value is FALSE. If set as TRUE, function will return a network diagram with nodes as estimated substructure proportions and edges as degree of similarity between the given node pair.
N_reference	numeric vector of the sample sizes for each of the K reference groups; must be specified if network = "TRUE".
reference_colors	A character vector of length K that specifies the color each reference group node in the network plot. If not specified, this defaults to K random colors.

Value

A data frame with the following columns:

goodness.of.fit: scaled objective loss from slsqp() reflecting the fit of the reference data. Values between 0.5-1.5 are considered moderate fit and should be used with caution. Values greater than 1.5 indicate poor fit, and users should not perform further analyses using Summix.

iterations: number of iterations for SLSQP algorithm

time: time in seconds of SLSQP algorithm

filtered: number of genetic variants not used in the reference group mixture proportion estimation due to missing values.

K columns of mixture proportions of reference groups input into the function

Author(s)

Adelle Price, <adelle.price@cuanschutz.edu>

Hayley Wolff, <hayley.wolff@cuanschutz.edu>

Audrey Hendricks, <audrey.hendricks@cuanschutz.edu>

References

https://github.com/hendriau/Summix

See Also

https://github.com/hendriau/Summix for further documentation and https://github.com/hendriau/Summix2_manuscript for a larger sample data set and description of simulations in Summix2 manuscript. slsqp function in the nloptr package for further details on Sequential Quadratic Programming https://www.rdocumentation.org/packages/nloptr/versions/1.2.2.2/topics/slsqp

Examples

```
# load the data
data("ancestryData")

# Estimate 5 reference ancestry proportion values for the gnomAD African/African American group
# using a starting guess of .2 for each ancestry proportion.
summix(data = ancestryData,
    reference=c("reference_AF_afr",
        "reference_AF_eas",
        "reference_AF_eur",
        "reference_AF_iam",
        "reference_AF_sas"),
    observed="gnomad_AF_afr",
    pi.start = c(.2, .2, .2, .2, .2),
    goodness.of.fit=TRUE)
```

summix_calc *summix_calc*

Description

Helper function for estimating mixture proportions of reference groups from large (N SNPs>10,000) genetic AF data, using slsqp to solve for least square difference

Usage

```
summix_calc(data, reference, observed, pi.start = NA)
```

Arguments

data	A dataframe of the observed and reference allele frequencies for N genetic variants. See data formatting document at https://github.com/hendriau/Summix for more information.
reference	A character vector of the column names for the reference groups.
observed	A character value that is the column name for the observed group.
pi.start	Length K numeric vector of the starting guess for the reference group proportions. If not specified, this defaults to 1/K where K is the number of reference groups.

Value

data frame with the following columns

objective: least square value at solution

iterations: number of iterations for SLSQP algorithm

time: time in seconds of SLSQP algorithm

filtered: number of SNPs not used in estimation due to missing values

K columns of mixture proportions of reference groups input into the function

summix_local *summix_local*

Description

Estimates local substructure mixture proportions in genetic summary data; Also performs a selection scan (optional) that identifies potential regions of selection along the given chromosome.

Usage

```
summix_local(
  data,
  reference,
  observed,
  goodness.of.fit = TRUE,
  type = "variants",
  algorithm = "fastcatch",
  minVariants = 0,
  maxVariants = 0,
  maxWindowSize = 0,
  minWindowSize = 0,
  windowOverlap = 200,
  maxStepSize = 1000,
  diffThreshold = 0.02,
  NSimRef = NULL,
  override_fit = FALSE,
  override_removeSmallAnc = FALSE,
  selection_scan = FALSE,
  position_col = "POS",
  nSimSE = 1000
)
```

Arguments

data	a data frame of the observed group and reference group allele frequencies for N genetic variants on a single chromosome. Must contain a column specifying the genetic variant positions.
reference	a character vector of the column names for K reference groups.
observed	a character value that is the column name for the observed group.
goodness.of.fit	an option to override the default scaled objective to return the raw loss from slsqp
type	user choice of how to define window size; options "variants" and "bp" are available where "variants" defines window size as the number of variants in a given window and "bp" defines window size as the number of base pairs in a given window. Default is "variants".
algorithm	user choice of algorithm to define local substructure blocks; options "fastcatch" and "windows" are available. "windows" uses a fixed window in a sliding windows algorithm. "fastcatch" allows dynamic window sizes. The "fastcatch" algorithm is recommended- though it is computationally slower. Default is "fastcatch".
minVariants	Used if algorithm = "fastcatch" and type = "variants". A numeric value that specifies the minimum number of genetic variants allowed to define a given window.
maxVariants	Used if type = "variants". A numeric value that specifies the maximum number of genetic variants allowed to define a given window.

`maxWindowSize` Used if type = "bp". A numeric value that defines the maximum allowed window size by the number of base pairs in a given window.

`minWindowSize` Used if algorithm = "fastcatch" and type = "bp". A numeric value that specifies the minimum number of base pairs allowed to define a given window.

`windowOverlap` Used if algorithm = "windows". A numeric value that defines the number of variants or the number of base pairs that overlap between the given sliding windows. Default is 200.

`maxStepSize` a numeric value that defines the maximum gap in base pairs between two consecutive genetic variants within a given window. Default is 1000.

`diffThreshold` Used if algorithm = "fastcatch". A numeric value that defines the percent difference threshold to mark the end of a local substructure block. Default is 0.02.

`NSimRef` Used if f selection_scan = TRUE. A numeric vector of the sample sizes for each of the K reference groups that is in the same order as the reference parameter. This is used in a simulation framework that calculates within local substructure block standard error.

`override_fit` default is FALSE. If set as TRUE, the user will override the auto-stop of summix_local() that occurs if the global goodness of fit value is greater than 1.5 (indicating a poor fit of the reference data to the observed data).

`override_removeSmallAnc` default is FALSE. If set as TRUE, the user will override the automatic removal of reference ancestries with <2% global proportions – this is not recommended.

`selection_scan` user option to perform a selection scan on the given chromosome. Default is FALSE. If set as TRUE, a test statistic will be calculated for each local substructure block. Note: the user can expect extended computation time if this option is set as TRUE.

`position_col` a character value that is the column name for the genetic variants positions. Default is "POS".

`nSimSE` user choice of number of internal simulations to run to calculate standard error of estimates. Default is 1000.

Value

data frame with a row for each local substructure block and the following columns:

goodness.of.fit: scaled objective reflecting the fit of the reference data. Values between 0.5-1.5 are considered moderate fit and should be used with caution. Values greater than 1.5 indicate poor fit, and users should not perform further analyses using summix

iterations: number of iterations for SLSQP algorithm

time: time in seconds of SLSQP algorithm

filtered: number of SNPs not used in estimation due to missing values

K columns of mixture proportions of reference groups input into the function

nSNPs: number of SNPs in the given local substructure block

Author(s)

Hayley Wolff (Stoneman), <hayley.wolff@cuanschutz.edu>

Audrey Hendricks, <audrey.hendricks@cuanschutz.edu>

References

https://github.com/hendriau/Summix2

See Also

https://github.com/hendriau/Summix2 for further documentation.

Examples

```
data(ancestryData)
results <- summix_local(data = ancestryData,
                        reference = c("reference_AF_afr",
                                      "reference_AF_eas",
                                      "reference_AF_eur",
                                      "reference_AF_iam",
                                      "reference_AF_sas"),
                        NSimRef = c(704,787,741,47,545),
                        observed="gnomad_AF_afr",
                        goodness.of.fit = TRUE,
                        type = "variants",
                        algorithm = "fastcatch",
                        minVariants = 150,
                        maxVariants = 250,
                        maxStepSize = 1000,
                        diffThreshold = .02,
                        override_fit = FALSE,
                        override_removeSmallAnc = TRUE,
                        selection_scan = FALSE,
                        position_col = "POS")
print(results$results)
```

---

summix_network *summix_network*

Description

Helper function to plot the network diagram of estimated substructure proportions and similarity between reference groups

Usage

```
summix_network(
  data = data,
  sum_res = sum_res,
  reference = reference,
  N_reference = N_reference,
  reference_colors = reference_colors
)
```

Arguments

data	A dataframe of the observed and reference allele frequencies for N genetic variants. See data formatting document at https://github.com/hendriau/Summix for more information.
sum_res	The resulting data frame from the summix function
reference	A character vector of the column names for the reference groups.
N_reference	numeric vector of the sample sizes for each of the K reference groups.
reference_colors	A character vector of length K that specifies the color each reference group node in the network plot. If not specified, this defaults to K random colors.

Value

network diagram with nodes as estimated substructure proportions and edges as degree of similarity between the given node pair

---

testDiff *testDiff*

Description

Helper function to determine whether reference group has changed for fast/catchup window algorithm

Usage

```
testDiff(last, current, threshold = 0.01)
```

Arguments

last	substructure proportions of block returned from summix
current	substructure proportions of block returned from summix
threshold	if applicable the threshold for determining change point

Value

true if passes threshold, false if not

variantGetNext *variantGetNext*

Description

Helper function to get starting end point that is a minimum distance (in variants) from start point; uses indices NOT position numbers

Usage

```
variantGetNext(positions, start, minVariants)
```

Arguments

positions	list of positions of variants
start	index of the current start position
minVariants	integer defining the minimum size in number of variants of the window

Value

the new end point index

Index