Package: adwave (via r-universe)

October 15, 2024

Type Package

Title Wavelet Analysis of Genomic Data from Admixed Populations

Version 1.3

Date 2018-05-17

Depends waveslim

Author Jean Sanderson

Maintainer Murray Cox <murray.p.cox@gmail.com>

Description Implements wavelet-based approaches for describing population admixture. Principal Components Analysis (PCA) is used to define the population structure and produce a localized admixture signal for each individual. Wavelet summaries of the PCA output describe variation present in the data and can be related to population-level demographic processes. For more details, see J Sanderson, H Sudoyo, TM Karafet, MF Hammer and MP Cox. 2015. Reconstructing past admixture processes from local genomic ancestry using wavelet transformation. Genetics 200:469-481 <doi:10.1534/genetics.115.176842>.

URL https://doi.org/10.1534/genetics.115.176842

License GPL (>= 2)

NeedsCompilation no

Date/Publication 2018-05-17 09:15:46 UTC

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

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

RemoteRef HEAD

RemoteSha 4eb5e690bcdc91600bdeb5f7269a6c87a39f28c2

Contents

`admix` *Simulated Admixed Population Data*

Description

The example dataset provides simulated allele calls for 50 individuals from 3 populations. We consider a simple admixture scenario where two ancestral populations, popA (sample size n_A = 15) and popB (sample size n_B = 15), merged 160 generations ago to form the admixed population, popAB (sample size n_{AB} = 20). The ancestral populations contributed to the admixed population with probability 0.5.

Usage

```
data(admix)
```

Format

A list with the following entries:

- `data`: 3,000 x 50 data matrix with 3,000 genotype calls (rows) for 50 individuals (columns). Raw entries take the value 0 if heterozygous, and -1 or 1 if homozygous. Polymorphisms are ordered by their physical positions. Column names list individual IDs, row names list polymorphism IDs.
- `map`: 3,000 x 2 data matrix providing a mapping to genetic distance (‘recombination distance’) for each polymorphism.
- `colplot`: vector of length 50. Plotting color for each individual.
- `populations`: list of length 3. Character vectors giving the IDs of individuals in each population. IDs must map to the column names of the data matrix.

Details

Further description of the dataset can be found in Sanderson et al. (2015). The data were simulated using MaCS (Chen et al. 2009).

References

Sanderson J, H Sudoyo, TM Karafet, MF Hammer and MP Cox. 2015. Reconstructing past admixture processes from local genomic ancestry using wavelet transformation. *Genetics* 200:469-481. https://doi.org/10.1534/genetics.115.176842

Chen GK, P Marjoram and JD Wall. 2009. Fast and flexible simulation of DNA sequence data. *Genome Research* 19:136-142. https://doi.org/10.1101/gr.083634.108

Examples

```
data(admix)
str(admix)
```

`plotsignal` *Plot Localized Admixture Signals*

Description

Plotting function for objects of class `adsig`.

Usage

```
plotsignal(x, ind = NULL, popA = NULL, popB = NULL, xlab = NULL, ylab = NULL,
  ylim = NULL, main = NULL)
```

Arguments

`x`	object of class `adsig` for plotting.
`ind`	character giving ID of a single individual to plot.
`popA`	*(optional)* name of ancestral population 1.
`popB`	*(optional)* name of ancestral population 2.
`xlab`	*(optional)* character string for x axis label.
`ylab`	*(optional)* character string for y axis label.
`ylim`	*(optional)* vector giving plotting range for y axis.
`main`	*(optional)* character string for title.

Value

Produces figure.

Author(s)

Jean Sanderson

References

Sanderson J, H Sudoyo, TM Karafet, MF Hammer and MP Cox. 2015. Reconstructing past admixture processes from local genomic ancestry using wavelet transformation. *Genetics* 200:469-481. https://doi.org/10.1534/genetics.115.176842

See Also

`signal`

Examples

```
data(admix)

# Generate the admixture signal with windowing
AdexPCA2 <- signal(admix$data,popA="popA",popB="popB",populations=admix$populations,
  tol=0.001,n.signal=1000,window.size=0.01)

# Plot resulting admixture signal for one individual
plotsignal(AdexPCA2,ind="AD00001",popA=AdexPCA2$popA,popB=AdexPCA2$popB)
```

`signal` *Compute Localized Admixture Signals*

Description

Produces estimates of localized ancestry for each individual.

Usage

```
signal(table, who = colnames(table), populations, popA = NA, popB = NA,
 normalize = FALSE, n.pca = 5, PCAonly = FALSE, verbose = TRUE, tol = 0.001,
 n.signal = NULL, window.size = NULL, genmap = NULL)
```

Arguments

`table`	matrix of genotype calls (rows, length *T*) versus individuals (columns, length *n*).
`who`	individuals to include in the analysis.
`populations`	list containing a vector of IDs for each population in the analysis.
`popA`	name of ancestral population 1 (used for forming the axes of variation). Must match one of the names in `populations`.
`popB`	name of ancestral population 2 (used for forming the axes of variation). Must match one of the names in `populations`.
`normalize`	if `TRUE`, normalize the data matrix. Default is `FALSE`.
`n.pca`	number of PCA axes to compute (only the first principal component is used for forming the signals, but additional components may be desired for visualization). Default is 5.
`PCAonly`	if `TRUE`, only compute the PCA, do not compute the signals. Default is `FALSE`.
`verbose`	if `TRUE`, print summary to screen. Default is `TRUE`.
`tol`	tolerance for normalization of admixture signals (ϵ in accompanying paper). Default is 0.001.
`n.signal`	*(optional)* number of data points in the windowed signal.
`window.size`	*(optional)* size of window specified as a proportion of total length; e.g., `window.size = 0.01` with signal of length T = 3000 SNPs generates windows of $0.01x3000 = 30$ polymorphisms. Value need not be a round number.
`genmap`	*(optional)* genetic distance of genotype calls, supplied as vector of length *T*. If specified, signals will be formulated in terms of genetic distance along the chromosome (rather than physical position).

Details

Applies PCA to genome-wide data using ancestral reference populations. The first eigenvector reflects the population structure. All individuals are then projected on to this axis to form the SNP-level admixture signals. PCA scores are used to estimate the proportion of admixture at the level of individuals (`indP`) and populations (`popP`). There is no restriction on the length of the data (number of SNPs) and the default is to provide an estimate of localized ancestry at each SNP.

Optionally, it is also possible to window the signals, producing processed signals of length `n.signal`. The windows may be overlapping or disjoint with width specified through the `window.size` option (see examples). If `genmap` is specified, the signals will be formulated in terms of genetic distance along the chromosome (note: this function is not described in the accompanying paper).

Value

Returns an object of class `adsig`, a list with the following components:

`call`	function call.
`date`	date of function call.
`individuals`	individuals for whom projections on the first principal component are calculated.
`n.snps`	number of polymorphisms in the table.
`signals`	The admixture signals, output as a Txn data matrix, where *n* is the number of individuals and *T* is the number of data points (either the number of polymorphisms if `n.signal` = `NULL` or `n.signal` otherwise).
`n.tol`	the number of entries replaced by zero in the normalization procedure. This is dependent on the value set for the tolerance, tol.
`popP`	estimated proportion of admixture for each population.
`indP`	estimated proportion of admixture for each individual.
`pa.ind`	columns are principal axes in individual coordinates ($n_A + n_B$ rows, `n.pca` columns).
`pa.snp`	columns are principal axes in polymorphism coordinates (*T* rows, `n.pca` columns).
`G`	matrix of quadratic form in individual coordinates.
`ev`	vector of eigenvalues.
`gendist`	(*only if* `genmap` *is specified in input*) Vector of genetic distances along the chromosome, length `n.signal`.

Author(s)

Jean Sanderson

References

Sanderson J, H Sudoyo, TM Karafet, MF Hammer and MP Cox. 2015. Reconstructing past admixture processes from local genomic ancestry using wavelet transformation. *Genetics* 200:469-481. https://doi.org/10.1534/genetics.115.176842

See Also

wavesum, plotsignal

Examples

```
data(admix)

# EXAMPLE 1
# Generate the admixture signal
AdexPCA <- signal(admix$data,popA="popA",popB="popB",populations=admix$populations,tol=0.001,
  n.signal=NULL)

# Plot the resulting PCA
plot(AdexPCA$pc.ind[,1],AdexPCA$pc.ind[,2],col=admix$colplot,xlab="PC1",ylab="PC2",pch=16)
legend("bottomright",c("popA","popB","popAB"),col=c(3,4,2),pch=16)


# EXAMPLE 2
# Generate the admixture signal with windowing
AdexPCA2 <- signal(admix$data,popA="popA",popB="popB",populations=admix$populations,tol=0.001,
  n.signal=1000,window.size=0.01)

# Plot resulting admixture signal for one individual
plotsignal(AdexPCA2,ind="AD00001",popA=AdexPCA2$popA,popB=AdexPCA2$popB)


# EXAMPLE 3
# Generate the admixture signal with windowing
# As in EXAMPLE 2 but with n.signal reduced to 100 to provide disjoint windows
AdexPCA3 <- signal(admix$data,popA="popA",popB="popB",populations=admix$populations,tol=0.001,
  n.signal=100,window.size=0.01)

# Plot resulting admixture signal for one individual
plotsignal(AdexPCA3,ind="AD00001",popA=AdexPCA2$popA,popB=AdexPCA2$popB)


# EXAMPLE 4
# Generate the admixture signal in terms of genetic distance
# As in EXAMPLE 2 but with genmap specified so that signals are formulated using genetic distances
AdexPCA4 <- signal(admix$data,popA="popA",popB="popB",populations=admix$populations,tol=0.001,
 n.signal=1000,window.size=0.01,genmap=admix$map[,2])

# Plot resulting admixture signal for one individual
plotsignal(AdexPCA4,ind="AD00001",popA=AdexPCA4$popA,popB=AdexPCA4$popB)
```

wavesum *Wavelet Summaries of Localized Admixture Signals*

Description

Produces wavelet summaries for each individual and group. Returns the wavelet variance and average block size metric (ABS).

Usage

```
wavesum(x, populations, popA = NA, popB = NA, ml = NULL, type = "la8",
    t.factor = 1, fullWT = FALSE)
```

Arguments

`x`	object of class `adsig`.
`populations`	list containing a vector of individual IDs for each population in the analysis.
`popA`	name of ancestral population 1 (used for forming the axes of variation). Must match one of the names in `populations`.
`popB`	name of ancestral population 2 (used for forming the axes of variation). Must match one of the names in `populations`.
`ml`	number of wavelet scales in the decomposition. Must not exceed $log_2(T)$, where T is the length of the signal.
`type`	name of the wavelet to use in the decomposition. The default, “la8”, is Daubechies Least Asymmetric wavelet of length 8. Other options include “haar”.
`t.factor`	multiplicative factor for thresholding. See paper for details. Default is 1.
`fullWT`	if `TRUE`, save the full wavelet periodogram, as well as the wavelet variance. Object size will be large. Default is `FALSE`.

Details

Produces wavelet summaries for objects of class `adsig`. The function computes the wavelet variance for each individual and population, extracts the informative wavelet variance based on levels observed in the ancestral populations, and computes summary measures of average block size metric (ABS) and peak wavelet scale for each individual and population.

See `waveslim` documentation for details of the `modwt` function and alternative wavelet options.

Value

The code returns a list with the following components:

`n.ind`	number of individuals in the analysis.
`n.group`	number of groups in the analysis.
`rv.ind`	matrix of dimension $nxml$, returning the raw wavelet variance for each individual.
`rv.group`	matrix of dimension $n.groupxml$, returning the raw wavelet variance for each group.
`threshold`	vector of length ml, returning threshold values for each wavelet scale.

`iv.ind`	matrix of dimension $nxml$, returning the informative (thresholded) wavelet variance for each individual.
`iv.group`	matrix of dimension $n.groupxml$, returning the informative (thresholded) wavelet variance for each group.
`abs.ind`	vector of length n, returning the average block size metric (ABS) for each individual.
`abs.group`	vector of length `n.group`, returning the average block size metric (ABS) for each group.
`pws.ind`	vector of length n, returning the peak wavelet scale for each individual.
`pws.group`	vector of length `n.group`, returning the peak wavelet scale for each group.
`wtmatrix`	*(only if* `fullWT = TRUE`*)*. Array of dimension $Txnxml$, containing squared wavelet coefficients for each individual.
`wtmatrix.group`	*(only if* `fullWT = TRUE`*)*. Array of dimension $Txn.groupxml$, squared wavelet coefficients, averaged for each group.

Author(s)

Jean Sanderson

References

Sanderson J, H Sudoyo, TM Karafet, MF Hammer and MP Cox. 2015. Reconstructing past admixture processes from local genomic ancestry using wavelet transformation. *Genetics* 200:469-481. https://doi.org/10.1534/genetics.115.176842

See Also

`signal`

Examples

```
data(admix)

# Generate the admixture signal
AdexPCA <- signal(admix$data,popA="popA",popB="popB",populations=admix$populations,
  tol=0.001, n.signal=NULL)

# Compute wavelet summaries
WSN <- wavesum(AdexPCA,populations=admix$populations,popA="popA",popB="popB")

# Plot raw wavelet variance for each population
barplot(WSN$rv.group[3,],ylim=c(0,0.9),col="red", names.arg=1:11,border=NA)
barplot(WSN$rv.group[1,],ylim=c(0,0.9),col="green3",names.arg=1:11,border=NA,add=TRUE)
barplot(WSN$rv.group[2,],ylim=c(0,0.9),col="blue", names.arg=1:11,border=NA,add=TRUE)
legend("topright",c("popA","popB","popAB"),col=c(3,4,2),pch=15)
box()

# Plot informative wavelet variance for admixed population
barplot(WSN$iv.group[3,],ylim=c(0,0.15),col="red",names.arg=1:11,border=NA)
```

```
ABS <- round(WSN$abs.group[3],2)
text(11,0.13,paste("ABS=",ABS))
box()
```

Index