Package ‘PopGenReport’

July 21, 2025

Type Package

Title A Simple Framework to Analyse Population and Landscape Genetic Data

Version 3.1

Date 2023-10-11

Description Provides beginner friendly framework to analyse population genetic data. Based on 'adegenet' objects it uses 'knitr' to create comprehensive reports on spatial genetic data.
For detailed information how to use the package refer to the comprehensive tutorials or visit <http://www.popgenreport.org/>.

License GPL

Encoding UTF-8

LazyData true

Depends R (>= 3.0.0), knitr, adegenet (>= 2.0.0)

Imports lattice, RgoogleMaps, gap, calibrate, xtable, plyr, dismo, reshape2, ggplot2, R.utils, ade4, pegas, genetics, gdistance, vegan, sp, mmod, GGally, graphics, grDevices, methods, stats, utils, raster

Suggests sf

VignetteBuilder knitr

URL https://github.com/green-striped-gecko/PopGenReport

RoxygenNote 7.2.3

NeedsCompilation no

Author Bernd Gruber [aut, cre], Aaron Adamack [aut]

Maintainer Bernd Gruber <bernd.gruber@canberra.edu.au>

Repository CRAN

Date/Publication 2023-10-11 07:20:02 UTC

Contents

`addline` *Function to add lines to landscape*

Description

adds a line from x1 to x2 to a raster layer

Usage

```
addline(r, x1, x2, val, plot = FALSE)
```

Arguments

`r`	raster that represents the landscape
`x1`	from coordinates
`x2`	to coordinates
`val`	resistance value 1 is no resistance
`plot`	switch if landscape should be plotted

Value

returns modified raster layer

`addpoly` *Function to add a polygon to a landscape*

Description

adds a polygon to a raster layer

Usage

```
addpoly(r, pol, val, plot = TRUE)
```

Arguments

`r`	raster that represents the landscape
`pol`	coordinates of the polygon
`val`	resistance value, One equals no resistance
`plot`	switch if landscape should be plotted

Value

returns modified raster layer

allel.rich *Calculates the allelic richness for a genind object*

Description

The function calculates the allelic richness for each combination of population and locus for a genind object. To account for differences in sample sizes and genotyping success, rarefication is used in the calculation. The sample size for each combination of population and locus was set equal to the smallest number of alleles seen in a sample across all combinations of population and locus. Allelic richness was calculated using the methods of Mousadik and Petit (1996) which are in turn based upon the work of Hurlbert (1971).

Usage

```
allel.rich(population, min.alleles = NULL)
```

Arguments

`population`	a `genind` object (from package adegenet)
`min.alleles`	the minimum number of alleles that will be sampled for calculating allelic richness. If min.alleles is set to NULL the min.alleles sampled will be determined automatically (see description)

Details

This function is similar to the allelic.richness function in hiefstat. The main differences between the two packages are that allel.rich works on a genind object while allelic richness works on a data frame and allel.rich is capable of determining allelic richness for species with most ploidies while allelic.richness only works for haploid and diploid species.

Value

Returns a list with the following entries:

all.richness is the allelic richness for each combination of population and locus sum.richness is the sum of the allelic richnesses for each population mean.richness is the mean allelic richness across all loci alleles.sampled is the smallest number of individuals sampled across all combinations of population and locus multiplied by the ploidy of the species. pop.sizes is a matrix with the total number of alleles counted for each combination of population and locus.

Author(s)

Aaron Adamack, aaron.adamack@canberra.edu.au

References

El Mousadik A, Petit RJ. (1996) High level of genetic differentiation for allelic richness among populations of the argan tree [Argania spinosa (L.) Skeels] endemic to Morocco

See Also

`popgenreport`

Examples

```
 data(bilby)
 popgenreport(bilby, mk.allel.rich=TRUE, mk.pdf=FALSE)
 #to get a pdf output you need to have a running Latex version installed
#on your system.
#popgenreport(bilby, mk.allel.rich=TRUE, mk.pdf=TRUE)

data(bilby)
allel.rich(bilby)
```

`allele.dist` *Counts and visualises allele frequencies across loci and subpopulations*

Description

Counts the number of observations for each combination of allele variant and subpopulation for each locus. Calculates relative allele proportions for each subpopulations and then produces a heatmap using that data.

Usage

```
allele.dist(population, mk.figures = TRUE)
```

Arguments

`population`	this is the `genind` object the analysis will be based on
`mk.figures`	if set to FALSE no figures are plotted. Default is TRUE.

Value

Produces heatmaps of the relative allele frequencies for each subpopulation at each locus and returns a list containing the counts (count) for each combination of allele and subpopulation and the relative frequencies of alleles by subpopulation (frequency) for each locus. The color bars on the heatmaps shows the relative frequency of an allele within a subpopulation for a locus while the histogram gives an overall count for the number of combinations of allele and subpopulation with a relative frequency.

Author(s)

Aaron Adamack, aaron.adamack@canberra.edu.au

See Also

`popgenreport`

Examples

```
## Not run:
 data(bilby)
 #here we use only the first 50 individuals to speep up the example
 popgenreport(bilby, mk.allele.dist=TRUE, mk.pdf=FALSE)

#to get a pdf output you need to have a running Latex version installed on your system.
popgenreport(bilby, mk.allele.dist=TRUE, mk.pdf=TRUE)

## End(Not run)
```

`bilby` *Bilby data set*

Description

This is a synthetic sample data set (in `genind` format) of microsatellite data.

Usage

```
bilby
```

Format

genlight object

Author(s)

Bernd Gruber <`bernd.gruber@canberra.edu.au`>

`costdistances` *Calculates cost distances for a given landscape (resistance matrix)*

Description

calculates a cost distance matrix, to be used with run.popgensim

Usage

```
costdistances(landscape, locs, method, NN)
```

Arguments

`landscape`	a raster object coding the resistance of the landscape
`locs`	coordinates of the subpopulations
`method`	defines the type of cost distance, types are "least-cost", "rSPDistance" or "commute (Circuitscape type)"
`NN`	number of next neighbours recommendation is 8

Value

a costdistance matrix between all pairs of locs

`emigration` *Function to execute emigration on a pops object*

Description

emigration process on all population in one go

Usage

```
emigration(xp, perc.mig, emi.m, emi.table = NULL)
```

Arguments

`xp`	all pops combined in a list
`perc.mig`	percentage if migrating individuals
`emi.m`	emigration probability (normally based on cost dispersal distance)
`emi.table`	a fixed number of migrating individuals can be specified (overrides emi.m)

Value

a list, first entry are updated pops, second entry the number of disperserin a matrix

---

`gd.kosman` *Individual genetic distance calculation based on Kosman & Leonhard 2005*

Description

Calculates pairwise genetic distances between all individuals using the individual genetic distance measure of Kosman and Leonard (2005). This function is similiar to the dist.codom in the package mmod. The two functions differ in their treatment of individuals with missing data. dist.codom omits individuals from the calculation of pairwise individual genetic distances while this function includes individuals with missing data. This is done by simply calculating the mean individual pairwise genetic distance over all loci for which there are values. Note that depending on your computers capabilities, you may run into memory errors when this function is used on datasets with large numbers of individuals (>1000). Additionally, the time for this function to run can be lengthy. Using a PC with a 3.5 GHz processor to calculate pairwise genetic distances between individuals with 36 diploid loci it took 0.3 seconds for 100 individuals, 5 seconds for 500 individuals, 21 seconds for 1000 individuals, 84 seconds for 2000 individuals, and 194 seconds for 3000 individuals.

Usage

```
gd.kosman(population)
```

Arguments

`population`	this is the `genind` object the analysis will be based on.

Value

Returns a list with two distance matrices. The first (geneticdist) contains pairwise individual genetic distances for each individual within a population, the second (loci_used) provides the number of loci that were used to calculate pairwise individual genetic distances between a pair of individuals.

Author(s)

Aaron Adamack, aaron.adamack@canberra.edu.au

References

Kosman E., Leonard K.J. 2005. Similarity coefficients for molecular markers in studies of genetic relationships between individuals for haploid, diploid, and polyploidy species. Molecular Ecology 14:415-424.

See Also

`popgenreport`

Examples

```
## Not run:
data(bilby)
popgenreport(bilby, mk.gd.kosman = TRUE, mk.pdf=FALSE)

## End(Not run)
#to get a pdf output you need to have a running Latex version installed on your system.
#popgenreport(bilby, mk.gd.kosman = TRUE, mk.pdf=TRUE)
```

`gd.smouse` *Individual genetic distance calculation based on Smouse and Peakall 1999*

Description

Calculate pairwise genetic distances between all individuals using the individual genetic distance measure of Smouse and Peakall (1999). This function should produce the same results as the individual genetic distances calculated using GenAlEx and choosing the interpolate missing data option. Note that depending on your computers capabilities, you may run into memory errors when this function is used on datasets with large numbers of individuals (>1000). Additionally, the time for this function to run can be quite lengthy. Using a PC with a 3.5 GHz processor to calculate pairwise genetic distances between individuals with 36 diploid loci it took 2 seconds for 100 individuals, 51 seconds for 500 individuals, 200 seconds for 1000 individuals, 836 seconds (~14 minutes) for 2000 individuals, and 1793 seconds (~30 minutes) for 3000 individuals. Please note that for each of your population groupings there must be at least one individual with genotyped alleles for each locus (it doesn't have to be the same individual for every locus). If a population doesn't meet this requirement, it will be dropped from the analysis and a warning message will be generated.

Usage

```
gd.smouse(population, verbose = TRUE)
```

Arguments

`population`	this is the `genind` object that the analysis will be based on.
`verbose`	information on progress. For large data sets this could be informative as it allows the user to estimate the amount of time remaining until the function is complete. The counter shows the number of the population for which genetic distances are currently being determined.

Value

Returns pairwise individual genetic distances for each individual within a population.

Author(s)

Aaron Adamack, aaron.adamack@canberra.edu.au

References

Smouse PE, Peakall R. 1999. Spatial autocorrelation analysis of individual multiallele and multilocus genetic structure. Heredity 82: 561-573.

See Also

`popgenreport`

Examples

```
## Not run:
data(bilby)
popgenreport(bilby, mk.gd.smouse = TRUE, mk.pdf=FALSE)
```

```
#to get a pdf output you need to have a running Latex version installed on your system.
#popgenreport(bilby, mk.gd.smouse = TRUE, mk.pdf=TRUE)

## End(Not run)
```

genleastcost *Least-cost path analysis based on a friction matrix*

Description

This function calculates the pairwise distances (Euclidean, cost path distances and genetic distances) of populations using a friction matrix and a spatial genind object. The genind object needs to have coordinates in the same projected coordinate system as the friction matrix. The friction matrix can be either a single raster of a stack of several layers. If a stack is provided the specified cost distance is calculated for each layer in the stack. The output of this function can be used with the functions wassermann or lgrMMRR to test for the significance of a layer on the genetic structure.

Usage

```
genleastcost(
  cats,
  fric.raster,
  gen.distance,
  NN = NULL,
  pathtype = "leastcost",
  plotpath = TRUE,
  theta = 1
)
```

Arguments

cats	a spatial genind object. see ?popgenreport how to provide coordinates in genind objects
fric.raster	a friction matrix
gen.distance	specification which genetic distance method should be used to calculate pairwise genetic distances between populations ("D", "Gst.Nei", "Gst.Hedrick") or individuals ("Smouse", "Kosman", "propShared")
NN	Number of neighbours used when calculating the cost distance (possible values 4,8 or 16). As the default is NULL a value has to be provided if pathtype='leastcost'. NN=8 is most commonly used. Be aware that linear structures may cause artefacts in the least-cost paths, therefore inspect the actual least-cost paths in the provided output.
pathtype	Type of cost distance to be calculated (based on function in the gdistance package. Available distances are 'leastcost', 'commute' or 'rSPDistance'. See functions in the gdistance package for futher explanations. If the path type is set to 'leastcost' then paths and also pathlength are returned.

`plotpath`	switch if least cost paths should be plotted (works only if pathtype='leastcost'. Be aware this slows down the computation, but it is recommended to do this to check least cost paths visually.
`theta`	value needed for rSPDistance function. see `rSPDistance` in package `gdistance`.

Details

to be written

Value

returns a list that consists of four pairwise distance matrixes (Euclidean, Cost, length of path and genetic) and the actual paths as spatial line objects.

Author(s)

Bernd Gruber

References

Cushman, S., Wasserman, T., Landguth, E. and Shirk, A. (2013). Re-Evaluating Causal Modeling with Mantel Tests in Landscape Genetics. Diversity, 5(1), 51-72.

Landguth, E. L., Cushman, S. A., Schwartz, M. K., McKelvey, K. S., Murphy, M. and Luikart, G. (2010). Quantifying the lag time to detect barriers in landscape genetics. Molecular ecology, 4179-4191.

Wasserman, T. N., Cushman, S. A., Schwartz, M. K. and Wallin, D. O. (2010). Spatial scaling and multi-model inference in landscape genetics: Martes americana in northern Idaho. Landscape Ecology, 25(10), 1601-1612.

See Also

`landgenreport`, `popgenreport`, `wassermann`, `lgrMMRR`

Examples

```
library(raster)
fric.raster <- readRDS(system.file("extdata","fric.raster.rdata", package="PopGenReport"))
glc <- genleastcost(cats=landgen, fric.raster, "D", NN=8)
wassermann(eucl.mat = glc$eucl.mat, cost.mat = glc$cost.mats, gen.mat = glc$gen.mat)
lgrMMRR(gen.mat = glc$gen.mat, cost.mats = glc$cost.mats, eucl.mat = glc$eucl.mat)
```

init.popgensim *Initialise a spatial meta-population for a popgen simulation*

Description

This functions initialises a time-forward, agent-based and spatiallly explicit genetic meta-population simulation

Usage

```
init.popgensim(
  n.pops,
  n.ind,
  sex.ratio,
  n.loci,
  n.allels,
  locs = NULL,
  n.cov = 3
)
```

Arguments

`n.pops`	number of subpopulations
`n.ind`	number of individuals per subpopulation
`sex.ratio`	sex ratio of males and females
`n.loci`	number of loci
`n.allels`	number of maximal alleles per loci
`locs`	coordinates of the subpopulations, provided as a row named data.frame(x=, y=) with n.pops rows.[Only used to name the subpopulations by row.names(locs). If not provided subpopulations are simply numbered.
`n.cov`	number of covariates (currenlty do not change, defaults to 3. In future versions covariates such as age etc. will be implemented)

Details

To set up a population we have to specify the following parameters: n.pops defines the number of subpopulations and n.ind the number of individuals within subpopulations (carrying capacity). In the current implementaiton all subpopulations have the same number of individuals. sex.ratio determines the proportion of females in a subpopulation. finaly the number of loci and number of alleles need to be specified. locs is used to name the populations. For simplicity the names are provided via a data.frame of x and y coordinates (as they normally come from a genind object)

Value

a simpops object (to be used in run.popgensim), which is a list of subpopulations. Each subpopulation consists of a data.frame with a row for each individual (coding the covariates and genetic make-up).

See Also

`run.popgensim`

Examples

```
init.popgensim(n.pops = 5, n.ind=8, sex.ratio = 0.25, n.loci = 4, n.allels = 7, n.cov = 3)
```

`landgen` *A simulated genind data set with spatial coordinates*

Description

This data set is used to demonstrate the use of the `landgenreport` and `genleastcost` functions. It is a simple spatial `genind` object with 100 individuals in 10 populations and 20 loci with up to 20 alleles per loci.

Usage

```
landgen
```

Format

An object of class `genind` of length 1.

Author(s)

Bernd Gruber <`bernd.gruber@canberra.edu.au`>

See Also

`landgenreport`, `genleastcost`

Examples

```
data(landgen)
summary(landgen)
```

`landgenreport` *Create a landscape genetic report*

Description

This function is the landscape genetic version of the popgenreport function. It needs to be provided with a genind object with spatial coordinates, a friction map (raster) and a specification which type of genetic distance should be used. Once all three type of input are provided with the necessary input, a landscape genetic analysis using least cost path analysis is computed (see Cushman et al. 2010, Landguth et al. 2010). Depending on the genetic distance meassurement this is done on a subpopulation basis (D, Gst.Hedrick, Gst.Nei=Fst) or on an individual basis (Kosman, Smouse).

Usage

```
landgenreport(
  cats,
  fric.raster,
  gen.distance = "Gst.Nei",
  NN = NULL,
  pathtype = "leastcost",
  plotpath = TRUE,
  theta = 1,
  mk.resistance = TRUE,
  mapdotcolor = "blue",
  mapdotsize = 1,
  mapdotalpha = 0.4,
  mapdottype = 19,
  mapzoom = NULL,
  mk.custom = FALSE,
  fname = "LandGenReport",
  foldername = "results",
  path.pgr = NULL,
  mk.Rcode = FALSE,
  mk.complete = FALSE,
  mk.pdf = TRUE
)
```

Arguments

`cats`	a `genind` object with spatial coordinates in the other slot
`fric.raster`	friction (resistance) raster, that specifies the landscape where the analysis should be computed on. If fric.raser is a stack a cost distances are calculated for each layer in the stack.
`gen.distance`	type of genetic distance that should be used. Depending on the genetic distance meassurement this is done on a subpopulation basis (D, Gst.Hedrick, Gst.Nei=Fst) or on an individual basis (Kosman, Smouse, propShared). propShared is the proportion of shared alleles between individuals.

`NN`	Number of neighbours used when calculating the cost distance (possible values 4,8 or 16). As the default is NULL a value has to be provided if pathtype is 'leastcost'. NN=8 is most commonly used as it avoids a directional bias, but be aware that linear structures may cause artefacts in the least-cost paths in the NN=8 case, therefore we strongly recommend to inspect the actual least-cost paths in the provided output.
`pathtype`	Type of cost distance to be calculated (based on function in the `gdistance` package. Available distances are 'leastcost', 'commute' or 'rSPDistance'. See functions in the gdistance package for futher explanations.
`plotpath`	switch if least cost paths should be plotted (works only if pathtype='leastcost'. Be aware this slows down the computation, but it is recommended to check least cost paths visually.
`theta`	value needed for rSPDistance function. see `rSPDistance` in package `gdistance`.
`mk.resistance`	switch to do the landscape genetic analysis based on resistance matrices, should be set to TRUE
`mapdotcolor`	see `popgenreport`
`mapdotsize`	see `popgenreport`
`mapdotalpha`	see`popgenreport`
`mapdottype`	see `popgenreport`
`mapzoom`	see `popgenreport`
`mk.custom`	switch to add a customised part to the landgenreport
`fname`	see `popgenreport`
`foldername`	see `popgenreport`
`path.pgr`	see `popgenreport`
`mk.Rcode`	see `popgenreport`
`mk.complete`	see `popgenreport`
`mk.pdf`	see `popgenreport`

Details

Check the help pages of `popgenreport` how to include coordinates to a genind object. The coordinates need to be projected. Latlongs are not valid, because Euclidean distances are calcuated based on these coordinates. For an example how to convert latlongs into a projected format have a look at the vignette that comes with this package. The friction needs to be a raster and needs to be in the same projection as the genind object. Also the type of genetic distance to be used needs to be specified.

Value

Four distance matrices are returned. Pairwise Euclidean distances between subpopulations/individuals, cost distances, path lengths and genetic distances. Also following the approach of Wassermann et al. 2010 a series of partial mantel tests are performed. A multiple regression analysis based on Wang 2013 and Legendre 1994 is returned.The actual least-cost paths can be found under paths

Author(s)

Bernd Gruber (bernd.gruber@canberra.edu.au)

References

Cushman, S., Wasserman, T., Landguth, E. and Shirk, A. (2013). Re-Evaluating Causal Modeling with Mantel Tests in Landscape Genetics. Diversity, 5(1), 51-72.

Landguth, E. L., Cushman, S. A., Schwartz, M. K., McKelvey, K. S., Murphy, M. and Luikart, G. (2010). Quantifying the lag time to detect barriers in landscape genetics. Molecular ecology, 4179-4191.

Wang,I 2013. Examining the full effects of landscape heterogeneity on spatial genetic variation: a multiple matrix regression approach for quantifying geographic and ecological isolation. Evolution: 67-12: 3403-3411.

Wasserman, T. N., Cushman, S. A., Schwartz, M. K. and Wallin, D. O. (2010). Spatial scaling and multi-model inference in landscape genetics: Martes americana in northern Idaho. Landscape Ecology, 25(10), 1601-1612.

See Also

popgenreport, wassermann, genleastcost, lgrMMRR

Examples

```
library(raster)
fric.raster <- readRDS(system.file("extdata","fric.raster.rdata", package="PopGenReport"))
lc<-landgenreport(cats=landgen, fric.raster=fric.raster,
gen.distance="D", NN=4, mk.resistance=TRUE, mk.pdf=FALSE)
names(lc$leastcost)
```

lgrMMRR *Multiple Matrix Regression with Randomization analysis*

Description

performs Multiple Matrix Regression with Randomization analysis This method was implemented by Wang 2013 (MMRR function see references) and also by Sarah Goslee in package ecodist. lgrMMRR is a simple wrapper to have a more user friendly output.

Usage

```
lgrMMRR(gen.mat, cost.mats, eucl.mat = NULL, nperm = 999)
```

Arguments

`gen.mat`	a genetic distance matrix (e.g. output from `genleastcost`
`cost.mats`	a list of cost distance matrices
`eucl.mat`	pairwise Euclidean distance matrix. If not specificed ignored
`nperm`	the number of permutations

Details

Performs multiple regression on distance matrices following the methods outlined in Legendre et al. 1994 and implemented by Wang 2013.

Value

a table with the results of the matrix regression analysis. (regression coefficients and associated p-values from the permutation test (using the pseudo-t of Legendre et al. 1994). and also r.squared from and associated p-value from the permutation test. F.test.

Finally also the F-statistic and p-value for overall F-test for lack of fit.

Author(s)

Bernd Gruber (bernd.gruber@canberra.edu.au) using the implementation of Wang 2013.

References

Legendre, P.; Lapointe, F. and Casgrain, P. 1994. Modeling brain evolution from behavior: A permutational regression approach. Evolution 48: 1487-1499.

Lichstein, J. 2007. Multiple regression on distance matrices: A multivariate spatial analysis tool. Plant Ecology 188: 117-131.

Wang,I 2013. Examining the full effects of landscape heterogeneity on spatial genetic variation: a multiple matrix regression approach for quantifying geographic and ecological isolation. Evolution: 67-12: 3403-3411.

See Also

`popgenreport`, `genleastcost`, `landgenreport`, `wassermann`

Examples

```
data(landgen)
library(raster)
fric.raster <- readRDS(system.file("extdata","fric.raster.rdata", package="PopGenReport"))
glc <- genleastcost(landgen, fric.raster, "D", NN=4, path="leastcost")
lgrMMRR(glc$gen.mat, glc$cost.mats, glc$eucl.mat, nperm=999)
```

mutation *Function to execute mutation on a pop data.frame*

Description

mutation subprocess on single populations

Usage

```
mutation(x, n.allels, mutrate, mtype = "n.all", n.cov)
```

Arguments

`x`	a pop object
`n.allels`	number of alleles
`mutrate`	the mutation rate
`mtype`	the mutation process (currently on n.all is implemented. (=Choose any allel from 1:n.allels))
`n.cov`	(number of covariates, default)

Value

a dispersal probability matrix

Author(s)

Bernd Gruber, Erin Peterson

null.all *Checks for the presence of and determine the frequency of null alleles*

Description

The function null.all determines the frequency of null alleles at each locus of a genind object. As an initial step, the function makes a bootstrap estimate (based on the observed allele frequencies) of the probability of seeing the number of homozygotes observed for each allele. If there are a large number of null alleles present at a locus, it would result in multiple alleles at a locus having an excess of homozygotes. The second step of the function estimates the frequency of null alleles and a bootstrap confidence interval for each locus using the methods of Chakraborty et al. (1994) and Brookfield (1996). If the 95% confidence interval includes zero, it indicates that the frequency of null alleles at a locus does not significantly differ from zero.

Usage

```
null.all(population)
```

Arguments

population	a genind object (from the package adegenet)

Value

The function returns a list with two main components: homozygotes and null.allele.freq. Homozygotes contains the output of the first part of the analysis: determining the observed number of homozygotes for allele at each locus (homozygotes$observed), generating a distribution of the expected number of homozygotes for each allele at each locus (homozygotes$bootstrap) based upond the observed allele frequencies at a locus, and producing a summary table given the probability of observing the number of homozygotes (homozygotes$probability.obs). null.allele.freq list contains summary tables of the null allele frequency estimates based upon the forumulas of Chakraborty et al. (1994) (summary1), and Brookfield (1996) (summary2). For each summary table, the observed frequency is the null allele frequency determined from the observed heterozygosity and homozygosity at a locus. The median, 2.5th, and 97.5th percentiles are from bootstrap estimates of null allele frequencies obtained by resampling the individual genotypes from the original genind object.

Brookfield (1996) provides a brief discussion on which estimator should be used. In summary, it was recommended that Chakraborty et al. (1994)'s method (e.g. summary1) be used if there are individuals with no bands at a locus seen, but they are discounted as possible artefacts. If all individuals have one or more bands at a locus then Brookfield (1996)'s method (e.g. summary2) should be used.

Author(s)

Aaron Adamack, aaron.adamack@canberra.edu.au

References

Brookfield JFY. (1996) A simple new method for estimating null allele frequency from heterozygote deficiency. Molecular Ecology 5:453-455

Chakraborty R, Zhong Y, Jin L, Budowle B. (1994) Nondetectability of restriction fragments and independence of DNA fragment sizes within and between loci in RFLP typing of DNA. American Journal of Human Genetics 55:391-401

See Also

popgenreport

Examples

```
 ## Not run:
 data(bilby)
 #here we use only the first 50 individuals to speep up the example
 popgenreport(bilby, mk.null.all=TRUE, mk.pdf=FALSE)

#to get a pdf output you need to have a running Latex version installed on your system.
#popgenreport(bilby, mk.null.all=TRUE, mk.pdf=TRUE)

## End(Not run)
```

opt.landgen — *Function for optimising a landscape genetic analysis based on resistance layers*

Description

opt.landgen places iter times nlocations in a given landscape. For each scenario the pairwise cost-distances and Euclidean distances are calculated and the standard deviation of detour (see Gruber et al. in prep) is calculated. This metric evaluates the scenerio in their ability to find a significant effect for the given resistance layer on population structure (based on the causal modelling approach of Cushman et al.). To allow for more realistic designs previously locations, a minimal distance between locations and a mask that indicates "forbidden" areas can be specified.

Usage

```
opt.landgen(
  landscape,
  nlocations,
  mindist = 0,
  fixed = NULL,
  method,
  NN = 8,
  iter = 100,
  retry = 10,
  mask = NULL,
  plot = TRUE
)
```

Arguments

`landscape`	resistance layer as a raster object (if not projected dimensions are assumed to be in meters!!)
`nlocations`	the number of locations
`mindist`	minimal distance in map units (meter if not specified)
`fixed`	n fixed locations, provided as a data.frame with dimenstions nx2
`method`	least cost path algorithm provided by the gdistance package. Options are "least-cost", "SPDistance" and "commute". see function costdistances.
`NN`	number of next neighboring cells to be used, default is 8. see function costdistances.
`iter`	number of iterations that should be used to find an optimised design. Try initially the default and if it runs fast (depends on the type of costdistance an d dimenstions of landscape), increase to higher values.
`retry`	number of retries if optimisation is not possible in a certain iteration (due to mindist and/or fixed locations). Defaults to 10, which should be sufficient in most cases.

`mask`	a raster object that masks positions which are not available. Areas which are not to be used for locations are coded as NA (not available), all other values are treated as suitable for placing locations.
`plot`	if true, some plots are presented that show the histogramm, ecdf of the best (and the worst scenario).

Value

a list object with two slots. The first slot is called opt and contains iter optimisation values (sd.detour and sd.cost) in a iter x 2 dimenstional data.frame. The second slot is called scenario and contains the corrosponding 1:iter scenarios, given as coordinates in a data.frame of dimensions nlocations x 2. Both slots are ordered in decreasing order of sd.detour values. So the best scenario is at 1 and the worst is at position iter.

Examples

```
library(raster)
fric.raster <- readRDS(system.file("extdata","fric.raster.rdata", package="PopGenReport"))
opt.landgen(landscape = fric.raster, nlocations = 5, mindist = 3,
method = "leastcost", NN = 8, iter = 10)
```

`p2p` *Function to calculate dispersal distances based on cost distances*

Description

converts cost distances to probabilities: to reach a certain patch[d0 average distance of p of all individuals, for example d0=100, p =0.5 -> 50% procent of all migrating individuals go up to 100 m.

Usage

```
p2p(x, d0, p)
```

Arguments

`x`	Cost (Euclidean) distance matrix
`d0`	dispersal distance
`p`	of all individuals in a population

Value

a dispersal probability matrix

`pairwise.fstb` *Calculates pairwise fsts using a genind object (very fast)*

Description

for simulation the original pairwise.fst was too slow. The fast version works only with genind objects without NAs and diploid data (so to be save do not use it on non-simulated data)

Usage

```
pairwise.fstb(gsp)
```

Arguments

`gsp`	a genind object

Value

a pairwise fst matrix (same as hierfstat pairwise.fst)

`pairwise.propShared` *Calculates proportion of shared alleles per pairs of populations*

Description

Calculates proportion of shared alleles per pairs of populations based on the minima of allele frequency for each allel (then summed and averaged over loci). Returns a similarity matrix (upper diagonal of the pairwise matrix.

Usage

```
pairwise.propShared(gi)
```

Arguments

`gi`	a genind object with at least two populations

Value

a matrix of proportion of shared alleles between populations

`popgenreport` *This is the main function of the package. It analyses an object of class* `genind` *and then creates a report containing the results of the analysis. There are several routines that can be optionally included in the analysis and there are multiple output options including a PDF with the report, R-code and an object (*`fname.results`*) containing all of the results, which can be used for further analyses.*

Description

This function is used to analyse population genetic data. The main idea is to provide a framework for analysing microsatellite and also SNP genetic data (if not too many loci, say below 1000) using a mix of existing and new functions. The function works on an object of class `genind`. There are several ways to convert data into a `genind` object using existing functions provided by the `adegenet` package (`import2genind`, `df2genind`,`read.fstat`, `read.structure`, `read.genetix` ,`read.genepop`) or refer to `read.genetable` how to import data from an EXCEL (csv) document. The function performs a number of different genetic analyses (e.g. counts of indivuals and alleles across sub-populations, tests for heterozygosity and Hardy-Weinberg Equilibrium, differentiation statistics Fst, G'st, Jost's D, and genetic distance between individuals and populations), with users having the option to select which analysis routines are included in the report. To select a routine, the user simply turns on a switch e.g. mk.map=TRUE returns a map with the sampling location for each individual (if coordinates are provided).
Coordinates need to specified within the genind object. As a standard genind object does not require spatial coordinates, we extended it by using the `other` slot in the genind object. The easiest way to provide spatial coordinates is to use the read.genetable function and use the `lat`, `long` or `x`, `y` arguments for WGS1984 projected data or mercator projected data respectively. To calculate distances the data are internally reprojected using the `Mercator` function in package `dismo`), which is the projection used by google maps. Or you can add data manually to your `genind` object using the mentioned (e.g. `genindobject@other$latlong <- yourlatlong data` or `genindobject@other$xy <- your_xy_data`). If you have your data in a different projection you need to reproject them into either WGS1984 or the google maps Mercator projection. If you use a different projection distance calculation may be wrong and probably the map will not be correct. See the manual for an example how to project and add spatial coordinates to your genetic data.
Names for alleles (`genindobject@loc.names`) are truncated if longer than six characters. If truncated Captial letters linked by a hyphen are added to guarentee they are unique. You can rename them by providing new names by accessing the `genind@loc.names` slot prior to running `popgenreport`.
Note that the popgenreport function can take a long time to run if the options mk.complete, mk.gd.kosman, or mk.gd.smouse are set to `TRUE`. For example, running popgenreport with `mk.complete=TRUE` on a dataset with 500 individuals with 36 loci will take 14 to 15 minutes on a PC with a 3.5 Ghz processor and nearly 3 hours for a dataset with ~3200 individuals.

Usage

```
popgenreport(
  cats = NULL,
  mk.counts = TRUE,
  mk.map = FALSE,
  maptype = "satellite",
  mapdotcolor = "blue",
  mapdotsize = 1,
  mapdotalpha = 0.4,
  mapdottype = 19,
  mapzoom = NULL,
  mk.locihz = FALSE,
  mk.hwe = FALSE,
  mk.fst = FALSE,
  mk.gd.smouse = FALSE,
  mk.gd.kosman = FALSE,
  mk.pcoa = FALSE,
  mk.spautocor = FALSE,
  mk.allele.dist = FALSE,
  mk.null.all = FALSE,
  mk.allel.rich = FALSE,
  mk.differ.stats = FALSE,
  mk.custom = FALSE,
  fname = "PopGenReport",
  foldername = "results",
  path.pgr = NULL,
  mk.Rcode = FALSE,
  mk.complete = FALSE,
  mk.pdf = TRUE
)
```

Arguments

`cats`	this is the `genind` object the analysis will be based on.
`mk.counts`	switch is to provide overview counts of the number of individuals sampled, numbers of individuals and alleles sampled per sub-population, number of alleles per locus, mean number of alleles per locus and the percentatge of missing data.
`mk.map`	switch to produce a map with the sampling location of each individual marked. This switch requires individual coordinates (latitudes and longitudes in WGS1984) be provided (under cats@other$latlong or see `read.genetable` on how to import them from a table of genetic data). An error message will be generated if you turn this routine on, but do not provide the coordinates in the right format. If the coordinates are provided in a seperate file, they must be attached to the genind object in the slot `yourgenindobject@other$latlong <- yourlatlongdata`. `yourlatlongdata` needs to be a data frame that has the same number and order of individuals per row as the population genetic data. Note that an internet connection is required to connect to the Google Maps server which provides the basemap for this routine.
`maptype`	Defines the type of map. Default is 'satellite'. Other options are: 'roadmap', 'mobile', 'terrain', 'hybrid'.
`mapdotcolor`	Color of dots for each individual on the map. Default is 'blue'.

`mapdotsize`	Size of dots for each individual. Default is 1.
`mapdotalpha`	Transparency of dots. 1 is invisible, 0 is no transparency. Default is 0.4.
`mapdottype`	Defines the type of the symbol. For explanation see pch under `par`. Default is 19 - a filled circle.
`mapzoom`	Zoom level of the map. If not specified the default zoom of Google maps are used. Please be aware if you set the zoom level to high, the map may not show all sample locations.
`mk.locihz`	switch to test for population heterozygosity
`mk.hwe`	switch to test for Hardy-Weinberg equilibrium for each loci and population
`mk.fst`	switch to calculate Fst values for each loci and pairwise Fst (Nei's 1973) over subpopulations
`mk.gd.smouse`	Individual pairwise genetic distances based on Smouse and Peakall (1999). Refer to `gd_smouse`. Spatial coordinates need to be provided to be able to run this analysis.
`mk.gd.kosman`	Individual pairwise genetic distances based on Kosman & Leonhard (2005). Refer to `gd_kosman`. Spatial coordinates need to be provided to be able to run this analysis.
`mk.pcoa`	Principal component analysis following Jombart et al. 2009. Spatial coordinates need to be provided to be able to run this analysis. Refer to vignettes within `adegenet`.
`mk.spautocor`	Spatial autocorrelation analysis following Smouse & Peakall 1999. Spatial coordinates need to be provided to be able to run this analysis. Refer to `spautocor` for more information.
`mk.allele.dist`	switch to look at allele distributions by loci and subpopulation
`mk.null.all`	check for null alleles
`mk.allel.rich`	calculation of allelic richness
`mk.differ.stats`	switch to look at population differentiation statistics (Nei's Gst, Hedrick's Gst, and Jost's D)
`mk.custom`	edit custom.snw to include your own function to a report.
`fname`	filename for the output files. Defauts to PopGenReport. Note that using a filename which includes a space in the name will result in the filename for each figure being printed out in the PDF report for each figure. Replacing the space with an underscore should prevent this from happening.
`foldername`	name of folder, where files are stored. Defaults to 'results'
`path.pgr`	Folder where the output files are stored. Defaults to the temporary directory (`tempdir()`). If you want to store the output in another directory, simply provide the path here. e.g. `path.pgr=getwd()` saves it in your current working directory.
`mk.Rcode`	switch to get the full R script that is used to generate the report. A great way to get a very detailed insight on the kind of analysis and also an easy way to generate a script which you can customize for your analytical needs.

`mk.complete`	switch to create a full report using all of the routines (all switches are set to `TRUE`, except `mk.subgroups`).
`mk.pdf`	switch to create a shiny pdf output. You need a working **latex** version running on your system (e.g. MikTex (Windows) or Texmaker (Linux, MacOSX). For more information how to install latex on your system refer to the `http://www.popgenreport.org` and to the manuals of the `knitr` package and its manuals.

Value

The function returns an object (e.g. res) that has all of the results produced by this function in it. The structure of the object can be accessed via `str(res)`. The main slots in this object (if you ran a full report) are:
`dataoverview`, `PopHet`, `Alleledist`, `Fst`,`HsHtdifferentiate`, `HWEresults`, `subgroups`, `GDKosman`, `GDSmouse`

Additional ouput is provided in the form of a PDF (if mk.pdf=TRUE),which will be saved to the specified subfolder (via foldername) in your current working directory, and maps and figures which will be placed in this folder as well. This folder will be generated automatically in your current working directory. If you do not specify a working directory via `path.pgr` then the temporary working directory of R will be used (`tempdir()`). If `mk.Rcode=T` is set, an R file named fname.R will be saved to your specified subfolder.

Author(s)

Aaron Adamack & Bernd Gruber, aaron.adamack@canberra.edu.au, bernd.gruber@canberra.edu.au

References

Kosman E., Leonard K.J. 2005. Similarity coefficients for molecular markers in studies of genetic relationships between individuals for haploid, diploid, and polyploidy species. Molecular Ecology 14:415-424

Peakall R., Smouse P. 2012. GenAlEx 6.5: Genetic analysis in Excel. Population genetic software for teaching and research - an update. Bioinformatics 28:2537-2539

See Also

`adegenet`, `pegas`, `mmod`

Examples

```
## Not run:
data(bilby) # a generated data set
res <- popgenreport(bilby, mk.counts=TRUE, mk.map=TRUE, mk.pdf=FALSE)
#check results via res or use created tables in the results folder.

### RUN ONLY with a working Latex version installed
res <- popgenreport(bilby, mk.counts=TRUE, mk.map=TRUE, mk.pdf=TRUE, path.pgr="c:/temp")
#for a full report in a single pdf set mk.complete to TRUE

## End(Not run)
```

`pops2genind` *Function converts pops to a genind object*

Description

converts pops into genind (to calculate Fst etc.)

Usage

```
pops2genind(x, locs = NULL, n.cov = 3)
```

Arguments

`x`	pops object (a list of pop)
`locs`	named coordinates of locations
`n.cov`	number of covariates (defaults to 3)

Value

a spatial genind object

`possums` *A genlight object created via the read.genetable functions [possum data set from Sarre et al. 2015]*

Description

A genlight object created via the read.genetable functions [possum data set from Sarre et al. 2015]

Usage

```
possums
```

Format

genind object

Author(s)

Bernd Gruber <`bernd.gruber@canberra.edu.au`>

References

Sarre, S.D., Aitken, N., Adamack, A.T., Macdonald, A.J., Gruber, B. & Cowan, P. (2014). Creating new evolutionary pathways through bioinvasion: The population genetics of brushtail possums in New Zealand. Molecular Ecology, 23, 3419-3433.

Examples

```
possums
```

`read.genetable` *Function to convert textfiles into a genind object (the format required for popgenreport)*

Description

This function converts genetic data provided in a tabular 'comma separated value' (.csv) file into a genind object, the data format required to run PopGenReport. At the moment it works only for codominant markers (e.g. microsatellites). This function is based on df2genind from the `adegenet` package.

Usage

```
read.genetable(
  filename,
  pop = NULL,
  ind = NULL,
  lat = NULL,
  long = NULL,
  x = NULL,
  y = NULL,
  other.min = NULL,
  other.max = NULL,
  oneColPerAll,
  NA.char = NA,
  sep = NULL,
  ncode = NULL,
  ploidy = 2
)
```

Arguments

`filename`	the name of your file in .csv file format
`pop`	the column number if subpopulations are known, otherwise set to NULL
`ind`	the column number of an individual identifier, otherwise set to NULL
`lat`	the column number where the latitudinal coordinate is recorded (can be set to NULL)
`long`	the column number where the longitudinal coordinate is recorded (can be set to NULL)
`x`	the column number where the x coordinate is recorded. If not in Mercator it needs to be transformed!!! (can be set to NULL)

`y`	the column number where the y coordinate is recorded. If not in Mercator it needs to be transformed!!! (can be set to NULL)
`other.min`	if your data has some additional data (e.g. gender, size etc.) you can include this data in the genind object as well. Values in this section have to be in adjacent columns with no other columns between them. other.min is the column number where this section starts.
`other.max`	if your data has some additional data (e.g. gender, size etc.) then you can convert this data as well. This section has to be in a consecutive order with no other type of columns in between. other.max is the column number where this section ends.
`oneColPerAll`	needs to be specified. If your data is coded as a single column per allele
`NA.char`	can be NA, 0 or mean. See details section.
`sep`	a character string separating alleles. See details.
`ncode`	an optional integer giving the number of characters used for coding one genotype at one locus. If not provided, this is determined from data.
`ploidy`	Ploidy of the data set. Be aware that most analysis do only run for diploid data and that missing data in polyploid data sets are ambigious, which may give dubious results if not handled appropriately.

Details

The format of the .csv file is very important. Make sure your **headings are exactly as provided in the example file** or the conversion will likely fail (e.g. use lower cases for all headings). Use your favourite text editor to reformat the file (e.g. Excel) to prepare the data and save it as csv file. You need to provide the number of columns for each of your data sections. These are: ind, pop, lat, long, other.min, other.max, and whether there is a single column per allele (if you use a single column for both alleles then you need to specify the seperator as well), or two columns per allele. Please refer to the example files to make sure your file is in the correct format and then check the conversion by typing:

mydata <- read.genetable(\"mygeneticdat.csv\") mydata

The easiest way to provide spatial coordinates is to use the read.genetable function and use the `lat`, `long` arguments for WGS1984 projected data (the most common projection globally). For additional information how to use spatial data with PopGenReport refer to the help of the `popgenreport` function and to the popgenreport manual.
=== There are 3 treatments for missing values === - NA: kept as NA.

- 0: allelic frequencies are set to 0 on all alleles of the concerned locus. Recommended for a PCA on compositionnal data.

- \"mean\": missing values are replaced by the mean frequency of the corresponding allele, computed on the whole set of individuals. Recommended for a centred PCA.

=== Details for the sep argument === this character is directly used in reguar expressions like gsub, and thus require some characters to be preceeded by double backslashes. For instance, \"/\" works but \"|\" must be coded as \"\\|\".

Value

an object of the class genind This kind of object is needed to be passed on to the popgen.report function.

Author(s)

Bernd Gruber (bernd.gruber@canberra.edu.au)

See Also

import2genind, df2genind, read.fstat, read.structure, read.genetix read.genepop

Examples

```
#example file with one column per loci, seperated by forwardslash
read.csv(paste(.libPaths()[1],"/PopGenReport/extdata/platypus1c.csv", sep="" ))
platy1c <- read.genetable( paste(.libPaths()[1],"/PopGenReport/extdata/platypus1c.csv"
, sep="" ), ind=1, pop=2, lat=3, long=4, other.min=5, other.max=6, oneColPerAll=FALSE,
sep="/", )


#example file with two columns per loci
read.csv(paste(.libPaths()[1],"/PopGenReport/extdata/platypus2c.csv", sep="" ))
platy2c <- read.genetable( paste(.libPaths()[1],"/PopGenReport/extdata/platypus2c.csv",
 sep="" ), ind=1, pop=2, lat=3, long=4, other.min=5, other.max=6, oneColPerAll=TRUE)

#to get a pdf output you need to have a running Latex version installed on your system.
#run a report (with a map)
#res<- popgenreport(platy2c, mk.counts=TRUE, mk.map=TRUE, mk.allele.dist=TRUE, mk.pdf=TRUE)
```

---

reproduction *Function to execute reproduction on a pop data.frame*

Description

reproduction subprocess on single populations

Usage

```
reproduction(x, type = "K.limit", K = n.ind, n.off, n.cov)
```

Arguments

x	a pop object
type	type of density dependence K.limit only yet
K	Kapacity of a subpoplation
n.off	number of offspring per female
n.cov	(number of covariates, default 3)

Value

an updated pop object

run.popgensim *Run a time-forward popgen simulation*

Description

performs a time-forward, agent-based and spatiallly explicit genetic population simulation

Usage

```
run.popgensim(
  simpops,
  steps,
  cost.mat,
  n.offspring,
  n.ind,
  mig.rate,
  disp.max,
  disp.rate,
  n.allels,
  mut.rate,
  n.cov = 3,
  rec = "none",
  emi.table = NULL
)
```

Arguments

`simpops`	pops object (a list of pop)
`steps`	the number of steps (generations)
`cost.mat`	a cost matrix (e.g. calculated via costDistance)
`n.offspring`	number of offsprings per female
`n.ind`	number of individuals
`mig.rate`	migration rate
`disp.max`	dispersal distance of disp.rate individuals
`disp.rate`	percentage of individuals achieving disp.max
`n.allels`	number of maximal alleles at a loci
`mut.rate`	mutation rate
`n.cov`	number of covariates (defaults to 3)
`rec`	switch if emigration matrix should be recorded, either "none" or "emi"
`emi.table`	a emigration matrix, if provide a fixed number of migration events will take place otherwise based on disp.max, mig.rate and disp.rate,

Details

A pops object created via `init.popgensim` is used as input. The function simulates time forward individual-based spatially explicit population dynamics. Subpopulations are linked by dispersal that can be specified via a pairwise distance matrix between subpopulations [cost.mat]. Distances are converted to a probability. Currenlty the function used is the p2p function, where dispersal is modeled using an exponential function, that can be specified via disp.max and disp.rate. disp.max specifies the maximal distance that are achieved by the proportion of disp.rate individuals in a subpopulation. The number of dispersers per generation is set to round(mig.rate * n.ind). A simple mutation rate can be specified (the probability of a mutation per loci) using mut.rate. The maximal allowed number of alleles per loci need to be specified. Currently the mutation model is a simple Kmax-allele model [n.alleles]. As before n.cov is the number if covariates in the data.frame (currenlty fixed to n.cov=3). To track emigration events between subpopulations (be aware output is then a list instead of a simple pops object) rec can be set to "emi", which provides a matrix that shows the actual emigrations between subpopulations during the simulation. Emigration can also be determistic (instead of using disp.max and disp.rate) to a specified number of dispersal events per subpopulations. Over each generation events are occuring in that order: 1. dispersal, 2. reproduction, 3. mutation. For convinience the simulation can be run a specified number of generations [steps]. In case extra dynamics need to be modelled (e.g. one population is increased in number be a managment action or population are affected by environmental factors) simulations can also run only in single time steps [steps=1]. See example.

Value

an updated pops object after steps time steps or a list that includes the pops object and the emigration matrix [rec="emi"].

See Also

`init.popgensim`

Examples

```
library(raster)
set.seed(1)
locs <- cbind(x=round(runif(5,5,45)), y=round(runif(5,5,45)) )
cm <- as.matrix(dist(locs))
pops <- init.popgensim(n.pops = 5, n.ind=20, sex.ratio = 0.25, n.loci = 5, n.allels = 10, n.cov = 3)
#run pops
pops <- run.popgensim(pops, steps = 200, cost.mat= cm, n.offspring = 2, n.ind = 20,
mig.rate = 0.125, disp.max = 30, disp.rate =0.1, n.allels = 10, mut.rate = 0)
#convert to genind object
pops.gi <-pops2genind(pops)
#calculate pairwise fsts using pairwise.fstb
fsts <- pairwise.fstb(pops.gi)
#plot
plot(locs, xlim=c(0,50), ylim=c(0,50), pch=16,cex=4, col="darkgrey")
for (i in 1:4)
for (ii in (i+1):5)
lines(c(locs[i,1], locs[ii,1]), c(locs[i,2], locs[ii,2]), lwd=fsts[i,ii]*30, col="darkgreen")
text(locs+0.5, labels=1:5, col="white", font=2)
```

`spautocor` *Spatial autocorrelation following Smouse and Pekall 1999*

Description

Global spatial autocorrelation is a multivariate approach combining all loci into a single analysis. The autocorrelation coefficient r is calculated for each pairwise genetic distance pairs for all specified distance classes. For more information see Smouse and Peakall 1999, Peakall et a. 2003 and Smouse et al. 2008.

Usage

```
spautocor(gen.m, eucl.m, shuffle = FALSE, bins = 10)
```

Arguments

`gen.m`	a matrix of individual pairwise genetic distances. Easiest to use gd_smouse or gd_kosman to create such a matrix, but in priniciple any other squared distance matrix can be used. see example
`eucl.m`	A euclidean distance matrix, based on the coordinates of individuals. see example
`shuffle`	used internally for the permutation calculation
`bins`	number of bins for the distance classes. Currently only even bins are supported.

Value

Returns a data frame with r values and number of distances within each distance class.

Author(s)

Bernd Gruber, Bernd.Gruber@canberra.edu.au

References

Smouse PE, Peakall R. 1999. Spatial autocorrelation analysis of individual multiallele and multilocus genetic structure. Heredity 82: 561-573.

Double, MC, et al. 2005. Dispersal, philopatry and infidelity: dissecting local genetic structure in superb fairy-wrens (Malurus cyaneus). Evolution 59, 625-635.

Peakall, R, et al. 2003. Spatial autocorrelation analysis offers new insights into gene flow in the Australian bush rat, Rattus fuscipes. Evolution 57, 1182-1195.

Smouse, PE, et al. 2008. A heterogeneity test for fine-scale genetic structure. Molecular Ecology 17, 3389-3400.

Gonzales, E, et al. 2010. The impact of landscape disturbance on spatial genetic structure in the Guanacaste tree, Enterolobium cyclocarpum(Fabaceae). Journal of Heredity 101, 133-143.

Beck, N, et al. 2008. Social constraint and an absence of sex-biased dispersal drive fine-scale genetic structure in white-winged choughs. Molecular Ecology 17, 4346-4358.

See Also

popgenreport

Examples

```
## Not run:
data(bilby)
popgenreport(bilby, mk.spautocor=TRUE, mk.pdf=FALSE)
#to get a pdf output you need to have a running Latex version installed on your system.
#popgenreport(bilby[1:50], mk.spautocor=TRUE, mk.pdf=TRUE)

## End(Not run)
```

---

wassermann *Partial Mantel tests on costdistance matrices*

Description

This function implements the Causal modelling approach as suggested by Wassermann et al. 2010 and Cushman et al. 2010. It tests for the effect of landscape features using a cost distance matrix on the genetic structure of subpopulation/individuals.

Usage

```
wassermann(gen.mat, cost.mats, eucl.mat = NULL, plot = TRUE, nperm = 999)
```

Arguments

gen.mat	pairwise genetic distance matrix
cost.mats	pairwise cost distance matrix
eucl.mat	pairwise Eukclidean distance matrix
plot	switch for control plots of the partial mantel test
nperm	number of permutations for the partial mantel test

Details

see landgenreport

Value

A table with the results of the partial mantel test. Using plot=TRUE results in diagnostic plots for the partial mantel tests.

Author(s)

Bernd Gruber (bernd.gruber@canberra.edu.au)

References

Wassermann, T.N., Cushman, S. A., Schwartz, M. K. and Wallin, D. O. (2010). Spatial scaling and multi-model inference in landscape genetics: Martes americana in northern Idaho. Landscape Ecology, 25(10), 1601-1612.

See Also

`popgenreport`, `genleastcost`, `landgenreport`, `lgrMMRR`

Examples

```
library(raster)
fric.raster <- readRDS(system.file("extdata","fric.raster.rdata", package="PopGenReport"))
glc <- genleastcost(landgen, fric.raster, "D", NN=8)
wassermann(eucl.mat = glc$eucl.mat, cost.mats = glc$cost.mats, gen.mat = glc$gen.mat)
```

Index