

Package ‘kernelPop’

January 21, 2012

Type Package

Title Spatially explicit population genetic simulations

Version 0.11.2

Date 2005-09-04

Author Allan Strand and James Niehaus

Maintainer <stranda@cofc.edu>

Depends ape, ade4, MASS

Description This package creates a individual-based population genetic simulation. Individuals have spatial coordinates, dispersal governed by mixtures of Weibull and normal pdfs. It is discrete-time. It can be arbitrarily complex given enough known or assumed demographic characteristics. Simulates diploid and haploid inheritance. Allows the development of null distributions of genotypes for complex demographic scenarios. This version includes some approximations for pollen (male gamete) dispersal.

License GPL

Repository CRAN

Date/Publication 2010-01-26 08:26:43

R topics documented:

is.landscape	2
landscape.allelecount	3
landscape.allelefreq	4
landscape.amova	4
landscape.amova.locus	5
landscape.amova.pairwise	6
landscape.clean	6
landscape.coalinput	7

landscape.democol	8
landscape.demography	9
landscape.exp.het	9
landscape.Fst	10
landscape.FWright	11
landscape.genepop.output	12
landscape.generate.locations	12
landscape.locus	13
landscape.locus.states	14
landscape.locusvec	15
landscape.new.epoch	15
landscape.new.example	18
landscape.new.expression	18
landscape.new.floatparam	19
landscape.new.individuals	20
landscape.new.intparam	21
landscape.new.landscape	22
landscape.new.local.demo	23
landscape.new.locus	24
landscape.new.switchparam	25
landscape.obs.het	26
landscape.ploidy	26
landscape.plot.locations	27
landscape.pollen.kernel.demo	28
landscape.populations	28
landscape.sample	29
landscape.simulate	30
landscape.states	31
landscape.test.function	32
landscape.theta.h	32
landscape.theta.k	33
landscape.theta.s	34
landscape.to.rtheta	34
landscape.write.fdist	35
landscape.write.foreign	36
SimulationComponents	36

Index **38**

is.landscape	<i>Test whether an object is a (fairly) legitimate landscape</i>
--------------	--

Description

Test whether a genuine landscape

Usage

```
is.landscape(Rland = NULL, verb = TRUE, exact = FALSE)
```

Arguments

Rland	the Rmetasim landscape object
verb	print why not a landscape
exact	more strict

Examples

```
exampleland <- landscape.new.example()
is.landscape(exampleland)
rm(exampleland)
```

landscape.allelecount *Calculate allele numbers (frequency in the statistical sense) at each locus in each population*

Description

Calculate allele counts

Usage

```
hetmat <- landscape.exp.het(rland, tbl.out=F)
```

Arguments

rland	the Rmetasim landscape object
tbl.out	return as a (three-dimensional) table if TRUE. If FALSE, return as a dataframe with categorical variables denoting the locus, population and allele.

Value

Depends on the value of tbl.out. See above.

See Also

landscape.allelefreq, landscape.obs.het, landscape.exp.het, landscape.Fwright, landscape.Fst

Examples

```
# exampleland <- landscape.new.example()
# exampleland <- landscape.simulate(exampleland, 4)
# landscape.allelefreq(exampleland, tbl.out=TRUE)
# landscape.allelefreq(exampleland, tbl.out=FALSE)
# rm(exampleland)
```

landscape.allelefreq *Calculate allele frequencies at each locus in each population*

Description

Calculate allele frequencies

Usage

```
hetmat <- landscape.allelefreq(rland, tbl.out=F)
```

Arguments

rland the Rmetasim landscape object
tbl.out return as a (three-dimensional) table if TRUE. If FALSE, return as a dataframe with categorical variables denoting the locus, population and allele.

Value

Depends on the value of tbl.out. See above.

See Also

landscape.obs.het, landscape.exp.het, landscape.Fwright, landscape.Fst

Examples

```
# exampleland <- landscape.new.example()  
# exampleland <- landscape.simulate(exampleland, 4)  
# landscape.allelefreq(exampleland, tbl.out=TRUE)  
# landscape.allelefreq(exampleland, tbl.out=FALSE)  
# rm(exampleland)
```

landscape.amova *calculates phi-st for every locus in the landscape*

Description

calculates phi_ST for every locus in the landscape

Usage

```
landscape.amova(rland, np = 24, ns = 24)
```

Arguments

rland	landscape object
np	max number of pops to include
ns	max number of samples to collect

Value

vector of length equal to the number of loci

See Also

[landscape.amova.locus](#), [landscape.amova.pairwise](#)

landscape.amova.locus *uses functions in ade4 to calculate phi-st for a particular locus*

Description

Runs an amova on a locus. Does not include information about sequence similarity or ssr size in analysis.

Usage

```
landscape.amova.locus(l = 1, rland)
```

Arguments

l	locus number
rland	landscape object

Details

Should be the same as Weir and Cockerham's theta

Value

list of amova results for a locus

See Also

[landscape.amova](#), [landscape.amova.pairwise](#)

landscape.amova.pairwise

calculates pairwise phi_ST for a landscape

Description

pairwise phi_ST calculator. Kind of slow. use landscape.sample to reduce the size of the calculation.

Usage

```
landscape.amova.pairwise(rland)
```

Arguments

rland landscape object

See Also

landscape.amova, landscape.amova.locus

landscape.clean

Function to resolve inconsistencies within a landscape

Description

Converts a landscape to internal format and back. This can resolve inconsistencies in a 'hand-built' landscape

Usage

```
rland <- landscape.clean(rland)
```

Arguments

rland the Rmetasim landscape object

Examples

```
exampleland <- landscape.new.example()
exampleland <- landscape.simulate(exampleland, 4)
exampleland.clean <- landscape.clean(exampleland)
rm(exampleland)
```

landscape.coalinput *Add loci and individuals based upon output from SimCoal 2.0*

Description

Take rmetasim object and replaces the locus and individual data based on the results of a SimCoal run stored in Arlequin format files

Usage

```
## must be called AFTER integer, switch, and float params have been created
rland <- landscape.coalinput(rland, npp=500, arlseq = "seq.arp", arlms = "ms.arp", seqsitemut=1e-7,
                             msmut = 5e-4)
```

Arguments

rland	partially created landscape object, required
npp	number per population. Scalar or vector of length equal to number of populations. If scalar, value replicated
arlseq	name of the Arlequin format file containing a single locus of haploid sequence data for any number of populations
arlms	name of the Arlequin format file containing a single locus of diploid microsatellite data for any number of populations
seqsitemut	mutation rate for sequence data
msmut	mutation rate for diploid genotypic data

Details

This function provides part of an interface between R and SimCoal, an environment for simulating sequences and microsatellite genotypes from coalescent trees. SimCoal can be used to simulate a standing crop of alleles and their relationships under a wide range of demographies. It returns haplotypes and genotypes of individuals in Arlequin format files.

If either 'arlseq' or 'arlms' are set to NULL, their corresponding data will not be included in the landscape (for example if arlseq=NULL, only diploid genotypes will be imported)

The genotypes in the Arlequin files are used to create rland\$loci objects based upon their frequencies and states. These rland\$loci sub-objects are then used to populate the rland\$individuals sub-object.

The number of populations in the Arlequin files should be the same among genetic locus types (sequence versus microsatellite) and the rland\$intparam\$habitats parameter. The per-population frequency data will be used in creating individuals

Value

an rmetasim object with new loci and individuals

Author(s)

Mark Bravington and Allan Strand

Examples

```
exampleland <- landscape.new.empty()
exampleland <- landscape.new.intparam(exampleland, s=2, h=2)
exampleland <- landscape.new.floatparam(exampleland)
exampleland <- landscape.new.switchparam(exampleland)

# exampleland <- landscape.coalinput(exampleland)
# exampleland$loci
```

landscape.democol *return largest demographic column from a landscape*

Description

return largest demographic column from a landscape

Usage

```
landscape.democol()
```

Details

Useful to write functions that will be insensitive to some changes in the individuals object (mainly addition of non-genetic information)

Value

a scalar integer representing the largest column of demographic information in a landscape's individuals object

See Also

landscape.locus

landscape.demography *Calculate demographic parameters*

Description

Calculate demographic parameters from a landscape: CURRENTLY BROKEN!

Usage

```
rland <- landscape.demography(rland)
```

Arguments

rland the Rmetasim landscape object

Value

A list of length populations+1. The first 1..populations elements are lists comprised of lambda, the equilibrium stage-structure, the actual stage structure, a chi^2 value for the test of difference between predicted and actual, and an estimate of significance for that test. The last element of the main list is the same as the previous ones except it refers to the entire landscape

landscape.exp.het *Calculate expected heterozygosity*

Description

Calculate expected heterozygosity from a landscape

Usage

```
hetmat <- landscape.exp.het(rland)
```

Arguments

rland the Rmetasim landscape object

Details

Calculates the expected heterozygosity in each population:

$$1 - \sum_i p_i^2$$

where p is a vector of allele frequencies for a locus in a population.

Value

A matrix with num loci columns and num populations rows. Each element reflects the expected heterozygosity for that population x locus combination

See Also

landscape.obs.het, Fst.landscape

Examples

```
exampleland <- landscape.new.example()
exampleland <- landscape.simulate(exampleland, 4)
expHet <- landscape.exp.het(exampleland)
rm(exampleland)
```

landscape.Fst

Calculates population structure statistic for the entire landscape

Description

Calculate Fst for each allele at each locus in the landscape. If verb is set to TRUE, the function prints average Fst for loci and overall.

Usage

```
Fstmat <- landscape.Fst(rland, verb=F)
```

Arguments

rland the Rmetasim landscape object
verb determines whether there is verbose output

Details

Calculates Fst based upon the ratio of variance in allele frequency across subpopulations to the total variance in that allele's frequency. Does not calculate Wright's other statistics.

Value

A matrix with num alleles columns and num loci rows. Each element reflects the value of Fst for that allelexlocus combination. NA is assigned to alleles that are not present at a locus (either no longer or ever)

See Also

obs.het.landscape, exp.het.landscape, FWright.landscape

Examples

```
exampleland <- landscape.new.example()
exampleland <- landscape.simulate(exampleland, 4)
Fst <- landscape.Fst(exampleland, verb=TRUE)
Fst
rm(exampleland, Fst)
```

<code>landscape.FWright</code>	<i>Calculates Wright's Fixation index at each locus in each population</i>
--------------------------------	--

Description

Calculate F

Usage

```
Fmat <- landscape.FWright(rland)
```

Arguments

`rland` the Rmetasim landscape object

Value

A matrix of population x locus measures of Wright's F

See Also

`obs.het.landscape`, `exp.het.landscape`, `allelefreq.landscape`, `Fst.landscape`

Examples

```
exampleland <- landscape.new.example()
landscape.FWright(exampleland)
exampleland <- landscape.simulate(exampleland, 4)
landscape.FWright(exampleland)
rm(exampleland)
```

landscape.genepop.output

writes genepop files based on the genotypes of individuals in a landscape

Description

creates a genepop file. Occupied habitats correspond to genepop populations.

Usage

```
landscape.genepop.output(l, fn = "genepop.out", title = "rmetasim landscape output")
```

Arguments

l	landscape object
fn	output file name
title	title of output in genepop file

Details

Only exports diploid data. You might want to run `landscape.sample()` on the landscape before exporting it

See Also

`landscape.write.foreign`, `landscape.write.fdist`

Examples

```
# l <- landscape.new.example()
# l <- landscape.simulate(1,5)
# landscape.genepop.output(landscape.sample(1,3,24))
```

landscape.generate.locations

generates randomly located and sized, non-overlapping habitat patches on the landscape

Description

makes it easier to define landscapes, if you are just interested in a random landscape with certain qualities

Usage

```
landscape.generate.locations(npop = 10, xrange = c(0, 15000), yrange = xrange, sizekernel = c(300, 80))
```

Arguments

npop	number of habitats to define
xrange	the low and high bounds for a landscape's x coordinates (stay positive)
yrange	the low and high bounds for a landscape's y coordinates (stay positive)
sizekernel	mean and sd for a normal distribution defining the x-dimension of habitats
sizeykernel	mean and sd for a normal distribution defining the y-dimension of habitats
boundaries	another way to define boundaries for landscape.

Details

Chooses x and y coordinates for rectangles from a uniform distribution with limits given by the *range parameters. The x and y sizes of the habitats are governed by pulls from normal distributions

Value

a matrix with columns 1=left x coordinates, 2=bottom y coordinates, 3=right x coordinates, 4=top y coordinates. The row number is equal to the number of habitats (npop)

See Also

landscape.new.epoch()

landscape.locus	<i>return a matrix containing genotypes for a particular locus</i>
-----------------	--

Description

return a matrix containing genotypes for a particular locus

Usage

```
landscape.locus(lnum=1, Rland)
```

Arguments

lnum	the locus to return
Rland	the Rmetasim landscape object

Details

Returns a matrix with rows = `dim(rland$individuals)[1]`. The first three columns correspond to the class (and two placeholder variables) of an individual. Here rland is a landscape object. The remaining columns (1 if haploid, 2 if diploid) contain the allele indices for the various loci

Value

matrix

See Also

landscape.populations

Examples

```
exampleland <- landscape.new.example()
exampleland <- landscape.simulate(exampleland, 4)
print("Allele frequencies at locus 1")
table(landscape.locus(1,exampleland)[,c(-1:-(landscape.democol()))])
rm(exampleland)
```

landscape.locus.states

return a matrix containing actual allelic states and their indices

Description

Convenience function to return a matrix containing the states of the alleles and their indices for a particular locus

Usage

```
landscape.locus.states(lnum=1,Rland)
```

Arguments

lnum	the locus to return
Rland	the Rmetasim landscape object

Value

matrix

See Also

landscape.locus, landscape.states

landscape.locusvec *return a vector with the locus ids for each column in the individuals component of a landscape*

Description

return a vector with the locus ids for each column in the individuals component of a landscape

Usage

```
landscape.locusvec(Rland)
```

Arguments

Rland the Rmetasim landscape object

Value

vector

See Also

landscape.populations

Examples

```
exampleland <- landscape.new.example()
exampleland <- landscape.simulate(exampleland, 4)
landscape.locusvec(exampleland)
rm(exampleland)
```

landscape.new.epoch *Create an Epoch*

Description

Create an epoch for a Rmetasim landscape object

Usage

```
## must be called AFTER integer, switch, and float params have
## been created and after the demography has been created
## S, R, and M matrices must be square matrices of size X by X
## where X = rland$intparam$stages*rland$intparam$habitats
```

```
rland <- landscape.new.epoch(rland,S=NULL,R=NULL,M=NULL,epochprob=1,
                           startgen=0,extinct=NULL,carry=NULL,localprob=NULL,
                           pollen.kernels = NULL, seed.kernels = NULL,
                           leftx = NULL, rightx = NULL,
                           boty = NULL, topy = NULL, maxland = c(0, 0, 10000, 10000))
```

Arguments

rland	partially created landscape object, required
S	(default=NULL) Survivability matrix for epoch, NULL gives no movement between subpopulations (0 matrix)
R	(default=NULL) female Reproduction matrix for epoch, NULL gives no dispersal between subpopulations other than that determined by dispersal kernels(0 matrix)
M	(default=NULL) Male reproduction matrix for epoch, NULL gives no sperm or pollen movement between subpopulations other than that determined by dispersal kernels (0 matrix)
epochprob	(default=1) probability of choosing this epoch randomly if randepoch==1
startgen	(default=0) generation in which this epoch starts
extinct	(default=NULL) vector of extinction probabilities per generation for each subpopulation, must be rland\$intparam\$habitats in length, passing NULL gives a 0% probability of extinction to each subpopulation
carry	(default=NULL) vector of carrying capacities for each subpopulation, must be rland\$intparam\$habitats in length, passing NULL gives a 1000 individual carrying capacity to each subpopulation
localprob	(default=NULL) vector of probabilities for choosing local demographies, must be length(rland\$demography\$localdem) in length, passing NULL gives each demography an equal probability
leftx	vector of the left x-coordinates of habitat patches. If NULL (along with the other coordinates below), then the landscape is divided into a set of contiguous strips
rightx	vector of the right x-coordinates of habitat patches
topy	vector of the bottom y-coordinates of habitat patches
boty	vector of the top y-coordinates of habitat patches
pollen.kernels	A matrix that describes pollen kernels for each stage in the landscape. The rows correspond to stages in the landscape (there should be habitat * stages rows). The columns correspond to the characteristics of the dispersal kernels: Column 1 is the type of kernel (1=exponential, 2=Weibull, 3=mixture between Weibull and Gaussian) because the first is a special case of the second and the

second is a special case of the third type of dispersal kernel using "3" in this column is pretty much always reasonable. Columns 2 and 3 are scale and shape parameters of the Weibull (if shape=1, this reduces to an exponential kernel). Columns 4 and 5 correspond to the mean and sd of the Gaussian portion of the kernel. Column 6 is the mixture parameter, ranging from 0-1 that gives the relative weights of the Weibull versus Gaussian portions of the kernel. If the pollen kernel is constant across demographic stages, it is much easier to specify it when calling `landscape.new.floatparam()`

`seed.kernels` A similar matrix to `pollen.kernels`

Examples

```
exampleS <- matrix(c(0.1, 0, 0.5, 0.3), nrow = 2)
exampleR <- matrix(c(0, 1.1, 0, 0), nrow = 2)
exampleM <- matrix(c(0, 0, 0, 1), nrow = 2)

exampleland <- landscape.new.empty()
exampleland <- landscape.new.intparam(exampleland, s=2, h=2)
exampleland <- landscape.new.floatparam(exampleland)
exampleland <- landscape.new.switchparam(exampleland)
exampleland <- landscape.new.local.demo(exampleland,exampleS,exampleR,exampleM)

## nonsense matrices
exampleS <- matrix(c(rep(0,4),
                    rep(1,4),
                    rep(0,4),
                    rep(1,4)), nrow = 4)
exampleR <- matrix(c(rep(0.5,4),
                    rep(0,4),
                    rep(0.5,4),
                    rep(0,4)), nrow = 4)
exampleM <- matrix(c(rep(0,4),
                    rep(.25,4),
                    rep(0,4),
                    rep(0,4)), nrow = 4)

## defaults
exampleland<- landscape.new.epoch(exampleland,exampleS,exampleR,exampleM)

exampleland$demography$epochs[[1]]

rm(exampleS)
rm(exampleR)
rm(exampleM)
rm(exampleland)
```

landscape.new.example *Create a Default Landscape*

Description

Create a Rmetasim landscape with all default parameters.

Usage

```
rland <- landscape.new.example()
```

Arguments

None

Examples

```
## Only usage  
# landscape.new.example()
```

landscape.new.expression
don't use

Description

not implemented

Usage

```
landscape.new.expression(rland, expmat, hsq)
```

Arguments

rland	landscape object
expmat	expression matrix
hsq	narrow sense heritabilities

Details

This is only skeleton code.

Value

Not sure what it will return

Author(s)

Allan Strand

 landscape.new.floatparam

Create a set of floating point parameters

Description

Create a set of floating point parameters for a Rmetasim landscape.

Usage

```
## must be called AFTER landscape.new.empty()
rland <- landscape.new.floatparam(rland,s=0, seedscale = c(10,10),
  seedshape = c(10,10), seedmix=1, pollenscale=c(2,10),
  pollenshape=c(2,6), pollenmix=1,asp=1)
```

Arguments

rland	skeleton of landscape object, required
s	selfing (default=0), the selfing rate of the species
seedscale	two element vector that provides the scale of the exponential/Weibull and mean of the Gaussian components of the seed dispersal kernel.
seedshape	two element vector that provides the shape of the exponential/Weibull and sd of the Gaussian components of the seed dispersal kernel, respectively. If the first element of the vector is set to 1, the Weibull portion of the kernel becomes an exponential.
seedmix	the mixture parameter (ranges from 0-1) that weights the Weibull/exponential versus Gaussian portions of the dispersal kernel. A value of 1 selects dispersal distances from only the Weibull/exponential portion. A value of 0 selects distances only from the Gaussian portion.
pollenscale	two element vector that provides the scale of the exponential/Weibull and mean of the Gaussian components of the pollen dispersal kernel.
pollenshape	two element vector that provides the shape of the exponential/Weibull and sd of the Gaussian components of the pollen dispersal kernel, respectively. If the first element of the vector is set to 1, the Weibull portion of the kernel becomes an exponential.
pollenmix	the mixture parameter (ranges from 0-1) that weights the Weibull/exponential versus Gaussian portions of the dispersal kernel. A value of 1 selects dispersal distances from only the Weibull/exponential portion. A value of 0 selects distances only from the Gaussian portion.

Examples

```
## Defaults
exampleland <- landscape.new.empty()
exampleland <- landscape.new.floatparam(exampleland)
exampleland$floatparam

## .5 selfing rate
exampleland <- landscape.new.empty()
exampleland <- landscape.new.floatparam(exampleland,s=0.5)
exampleland$floatparam

rm(exampleland)
```

landscape.new.individuals

Fill a landscape with individuals

Description

Create a set of individuals for a Rmetasim landscape object.

Usage

```
## must be called AFTER integer, switch, and float params, demography,
## epochs, and loci have been created
```

```
rland <- landscape.new.individuals(rland,PopulationSizes)
```

Arguments

rland nearly complete landscape object, required

PopulationSizes vector of integers denoting how many individuals are in which stage and in which subpopulation, vector is ordered as: (pop1 stage1, pop1 stage2, ..., pop2 stage1, pop2stage2, ...), must be of length $rland\$intparam\$habitats * rland\$intparam\$stages$

Examples

```
exampleS <- matrix(c(0.1, 0, 0.5, 0.3), nrow = 2)
exampleR <- matrix(c(0, 1.1, 0, 0), nrow = 2)
exampleM <- matrix(c(0, 0, 0, 1), nrow = 2)

exampleland <- landscape.new.empty()
exampleland <- landscape.new.intparam(exampleland, s=2, h=2)
exampleland <- landscape.new.floatparam(exampleland)
exampleland <- landscape.new.switchparam(exampleland)
exampleland <- landscape.new.local.demo(exampleland,exampleS,exampleR,exampleM)
```

```

## nonsense matrices
exampleS <- matrix(c(rep(0,4),
                    rep(1,4),
                    rep(0,4),
                    rep(1,4)), nrow = 4)
exampleR <- matrix(c(rep(0.5,4),
                    rep(0,4),
                    rep(0.5,4),
                    rep(0,4)), nrow = 4)
exampleM <- matrix(c(rep(0,4),
                    rep(.25,4),
                    rep(0,4),
                    rep(0,4)), nrow = 4)

exampleland<- landscape.new.epoch(exampleland,exampleS,exampleR,exampleM)
exampleland <- landscape.new.locus(exampleland,type=2,ploidy=2,mutationrate=.001,numalleles=5,allelesize=100)
exampleland <- landscape.new.locus(exampleland,type=1,ploidy=1,mutationrate=.001,numalleles=3)
exampleland <- landscape.new.locus(exampleland,type=0,ploidy=2,mutationrate=.004,numalleles=4)

exampleland <- landscape.new.individuals(exampleland,
                                       c(5,20,7,15))

exampleland$individuals

rm(exampleS)
rm(exampleR)
rm(exampleM)
rm(exampleland)

```

landscape.new.intparam

Create a set of integer parameters

Description

Create a set of integer parameters for a Rmetasim landscape.

Usage

```

## must be called AFTER landscape.new.empty()
rland <- landscape.new.intparam(rland,h=2,s=1,cg=0,ce=0,totgen=500,maxland=10000)

```

Arguments

rland	skeleton of landscape object, required
h	habitats (default=1), the number of different subpopulations within the landscape
s	stages (default=1), the number of stages in the life cycle of the organism

cg	currentgen (default=0), the current generation the simulation has reached
ce	currentepoch (default=0), the current epoch the simulation has reached
totgen	totoalgens (default=1000), the total number of generations to simulate
maxland	maxlandsize(default=200000), the maxium number of individuals that can exist in the simulation

Examples

```
## Defaults
exampleland <- landscape.new.empty()
exampleland <- landscape.new.intparam(exampleland)
exampleland$intparam

## 2 habitats, 3 stage lifecycle, 1000000 generations, maximum 1000000 individuals
exampleland <- landscape.new.empty()
exampleland <- landscape.new.intparam(exampleland,h=2,s=2,totgen=1000000,maxland=1000000)
exampleland$intparam

rm(exampleland)
```

landscape.new.landscape

Create a Skeletal Landscape

Description

Create a skeletal Rmetasim landscape ready to be configured

Usage

```
rland <- landscape.new.empty()
```

Arguments

None

Examples

```
## Only usage
landscape.new.empty()
```

 landscape.new.local.demo

Create a Local Demography

Description

Create a local demography for an Rmetasim Landscape object

Usage

```
## must be called AFTER integer, switch, and float params have been created
## S, R, and M matrices must be square matrices of size rland$intparam$stages by rland$intparam$stages
rland <- landscape.new.local.demo(rland,S=Smatrix,R=Rmatrix,M=Mmatrix,k=0)
```

Arguments

rland	partially created landscape object, required
S	Survivability matrix for demography, required
R	female Reproduction matrix for demography, required
M	Male reproduction matrix for demography, required
k	flag for type of matrix, 0=demography at zero population density, 1=demography at carrying capacity

Details

The local demography objects encapsulate demography within a particular region. Multiple such objects can be defined to account for different demographies across space. The flag, k, can indicate whether the matrices represent demography at zero population growth and at carrying capacity, if density-dependence is modeled

Examples

```
exampleS <- matrix(c(0.1, 0, 0.5, 0.3), nrow = 2)
exampleR <- matrix(c(0, 1.1, 0, 0), nrow = 2)
exampleM <- matrix(c(0, 0, 0, 1), nrow = 2)

exampleland <- landscape.new.empty()
exampleland <- landscape.new.intparam(exampleland, s=2)
exampleland <- landscape.new.floatparam(exampleland)
exampleland <- landscape.new.switchparam(exampleland)
exampleland <- landscape.new.local.demo(exampleland,exampleS,exampleR,exampleM)

exampleland$demography$localdem

rm(exampleS)
rm(exampleR)
```

```
rm(exampleM)
rm(exampleland)
```

landscape.new.locus *Add a locus*

Description

Add a locus to a Rmetasim landscape object

Usage

```
## must be called AFTER integer, switch, and float params have been created
rland <- landscape.new.locus(rland,type=0,ploidy=1,mutationrate=0,transmission=1,numalleles=2,alle
```

Arguments

rland	partially created landscape object, required
type	(default=0) type of locus, 0=Infinite Allele mutation model (Integer), 1=Step-wise mutation model (Integer) state, 2=DNA base (variable length string state)
ploidy	(default=1) locus ploidy, 1 or 2
mutationrate	(default=0) probability of mutation per generation, less than or equal to 1
transmission	(default=1) 1=uniparental inheritance, 0=biparental inheritance
numalleles	(default=2) number of different alleles at the time of creation
allelesize	(default=50) length of DNA strings if type=2
frequencies	(default=NULL) vector of frequencies for each allele, must be numalleles long and add up to 1, if NULL frequencies are equally distributed

Examples

```
exampleland <- landscape.new.empty()
exampleland <- landscape.new.intparam(exampleland, s=2, h=2)
exampleland <- landscape.new.floatparam(exampleland)
exampleland <- landscape.new.switchparam(exampleland)

exampleland <- landscape.new.locus(exampleland,type=2,ploidy=2,mutationrate=.001,numalleles=5,allelesize=100)

exampleland$loci

rm(exampleland)
```

```
landscape.new.switchparam
```

Create a set of boolean parameters

Description

Create a set of boolean (1 or 0) parameters for a Rmetasim landscape.

Usage

```
## must be called AFTER landscape.new.empty()
rland <- landscape.new.switchparam(rland, re=1, rd=1, mp=1, dd=0)
```

Arguments

rland	skeleton of landscape object, required
re	randepoch (default=0), 1=randomly pick a new epoch (from the epochs listed in the landscape) after an epoch completes, 0=epochs are chosen in order
rd	randdemo (default=0), 1=randomly choose a demography (from the demographies listed in the landscape) for each subpopulation, 0=demographies are assigned in order
mp	multp (default=1), 1=multiple paternity, 0=entire families from a single mating
dd	density dependence. If dd=1, then two of each local demography matrix must be defined, the first set using new.local.demo with k=0 and representing demography at low density and again with k=1 for demography at high population density.

Examples

```
## Defaults
exampleland <- landscape.new.empty()
exampleland <- landscape.new.switchparam(exampleland)
exampleland$switchparam

## Random epochs, random demographies, and no multiple paternity
exampleland <- landscape.new.empty()
exampleland <- landscape.new.switchparam(exampleland, re=1, rd=1, mp=0)
exampleland$switchparam

rm(exampleland)
```

landscape.obs.het *Calculate observed heterozygosity*

Description

Calculate observed heterozygosity from a landscape

Usage

```
hetmat <- landscape.obs.het(rland)
```

Arguments

rland the Rmetasim landscape object

Value

A matrix with num loci columns and num populations rows. Each element reflects the observed heterozygosity for that population x locus combination

See Also

landscape.exp.het, landscape.Fst

Examples

```
# exampleland <- landscape.new.example()
# exampleland <- landscape.simulate(exampleland, 4)
# obshet <- landscape.obs.het(exampleland)
# rm(exampleland)
```

landscape.ploidy *return a vector with the ploidy of each locus*

Description

return a vector with the ploidy of each locus in the order they appear in the landscape

Usage

```
landscape.ploidy(Rland)
```

Arguments

Rland the Rmetasim landscape object

Value

vector

See Also

landscape.populations

Examples

```
exampleland <- landscape.new.example()
landscape.ploidy(exampleland)
rm(exampleland)
```

landscape.plot.locations

plots the locations of all habitats and individuals

Description

Plots the landscape. Gives different habitats different colors, though the colors cycle through rapidly when the number of habitats is large. Offspring always have the color of their *mother's* habitat. This provides a quick way to assess inter-habitat movement visually

Usage

```
landscape.plot.locations(rland, adults = c(NULL))
```

Arguments

rland	landscape
adults	vector of stages to select and plot, if NULL, plot all stages

Value

NULL

landscape.pollen.kernel.demo

takes parameters that define a pollen kernel and plots the results of simulating that kernel

Description

Produces a plot of a landscape. Adds a plot of the pollination distances in the landscape.

Usage

```
landscape.pollen.kernel.demo(s = 0, pmn1 = 30, pshp1 = 1.01, pmn2 = 600, pshp2 = 200, pmix = 0.75, plot =
```

Arguments

s	selfing rate to define
pmn1	mean of component distribution 1
pshp1	shape of component distribution 1
pmn2	mean of component distribution 2
pshp2	shape of component distribution 2
pmix	mixing parameter [0,1], 1=> all component 1
plot	boolean

Value

landscape object

Examples

```
l<-landscape.pollen.kernel.demo()
```

landscape.populations *return a vector of population IDs from a landscape*

Description

return a vector of population IDs from a landscape

Usage

```
landscape.populations(Rland)
```

Arguments

Rland	the Rmetasim landscape object
-------	-------------------------------

Details

Returns a vector of length `dim(rland$individuals)[1]` where `rland` is a landscape object. The vector classifies individuals into populations (or habitats)

Value

a vector

See Also

`landscape.locus`, `landscape.ploidy`

Examples

```
exampleland <- landscape.new.example()
exampleland <- landscape.simulate(exampleland, 4)
plot(table(landscape.populations(exampleland)),main="Distribution of population size in landscape")
rm(exampleland)
```

<code>landscape.sample</code>	<i>simulates sampling for genetics on the landscape</i>
-------------------------------	---

Description

Randomly pulls a max of `ns` individuals from a max of `np` populations and returns a landscape object that could be used for further simulation, but is usually used for analyses and summary statistics calculations

Usage

```
landscape.sample(rland, np = NULL, ns = NULL, pvec = NULL)
```

Arguments

<code>rland</code>	landscape object
<code>np</code>	number populations
<code>ns</code>	number samples per population
<code>pvec</code>	a vector of populations to sample. Should be numbers from 1 to number of habitats

Value

landscape object

Examples

```
l <- landscape.new.example()
l <- landscape.simulate(l,1)
l.samp <- landscape.sample(l,np=3,ns=24)
landscape.amova.pairwise(l.samp)
```

landscape.simulate *Run a simulation for a single landscape through time*

Description

Simulate a Rmetasim landscape for a number of generations.

Usage

```
rland <- landscape.simulate(rland=l,numit=100,seed=-1,compress=FALSE,adj.lambda=0)
```

Arguments

rland	the Rmetasim landscape object
numit	the number of generations/iterations to simulate, note that landscapes will not run past the rland\$intparam\$totalgens value
seed	The default value of seed uses the seed set in the calling environment. Any other value for seed uses 'set.seed()' to reset the random number generator. landscape.simulate uses the RNG selected by the calling environment.
compress	If true, landscape.simulate executes a survival and carrying capacity step before returning. In demographies with high reproductive potential, this can significantly reduce the size of R objects returned
adj.lambda	Tries to apply a correction to population growth that makes the observed growth rate more closely approximate that predicted from standard analysis eigensystem of the sum of the survival and reproduction Lefkovitch matrices

Examples

```
# exampleland <- landscape.new.example()
# exampleland <- landscape.simulate(exampleland, 4)
# exampleland
# rm(exampleland)
```

landscape.states	<i>return a list object containing actual allele states and their associated indices for a particular locus</i>
------------------	---

Description

return a list object containing actual allele states and their associated indices for a particular locus

Usage

```
landscape.states(lnum=1, Rland)
```

Arguments

lnum	the locus to return
Rland	the Rmetasim landscape object

Details

Returns a list with two elements, aindex containing the allele indices and state containing the actual allele states.

Value

list

See Also

landscape.locus, landscape.locus.states

Examples

```
exampleland <- landscape.new.example()
exampleland <- landscape.simulate(exampleland, 4)
landscape.states(1, exampleland)
rm(exampleland)
```

landscape.test.function
tests whatever c code is compiled in

Description

currently generates 10,000 random pulls from a pdf

Usage

```
landscape.test.function()
```

Value

vector

landscape.theta.h *Calculate theta using heterozygosity*

Description

Calculate theta from a landscape based upon heterozygosity.

Usage

```
theta.h.mat <- landscape.theta.h(rland)
```

Arguments

rland the Rmetasim landscape object

Details

Uses routines in the package 'ape'

Value

A matrix with num loci columns and num populations rows. Each element reflects the estimated theta for that population x locus combination

See Also

landscape.theta.k, landscape.theta.s

Examples

```
# exampleland <- landscape.new.example()
# exampleland <- landscape.simulate(exampleland, 4)
# landscape.theta.h(exampleland)
```

landscape.theta.k	<i>Calculate theta using the number of alleles</i>
-------------------	--

Description

Calculate theta using number of alleles from a landscape.

Usage

```
theta.k.mat <- landscape.theta.k(rland)
```

Arguments

rland the Rmetasim landscape object

Details

Uses routines in the package 'ape'

Value

A matrix with num loci columns and num populations rows. Each element reflects the estimated theta for that population x locus combination

See Also

landscape.theta.h, landscape.theta.s

Examples

```
# exampleland <- landscape.new.example()
# exampleland <- landscape.simulate(exampleland, 4)
# landscape.theta.k(exampleland)
```

landscape.theta.s *Calculate theta using segregating sites*

Description

Calculate theta from a landscape based upon the number of segregating sites.

Usage

```
theta.s.mat <- landscape.theta.s(rland)
```

Arguments

rland the Rmetasim landscape object

Details

Uses routines in the package 'ape'

Value

A matrix with num loci columns and num populations rows. Each element reflects the estimated theta for that population x locus combination

See Also

theta.k.landscape, theta.h.landscape

Examples

```
# exampleland <- landscape.new.example()
# exampleland <- landscape.simulate(exampleland, 4)
# theta.s.mat <- landscape.theta.s(exampleland)
# theta.s.mat
```

landscape.to.rtheta *Converts genetic marker data in a landscape into a format suitable for analysis using ANOVA*

Description

This function converts Rmetasim landscapes into a format that can be analyzed with ANOVA to calculate Weir's theta.

Usage

```
genin <- landscape.to.rtheta(Rland,numi)
```

Arguments

Rland	the Rmetasim landscape object
numi	number of individuals to sample at random from each population/habitat. The default value of 0 takes all individuals in each population. CAUTION this could take a long time.

Value

A matrix with columns representing the population, class, individual, locus, allelic position, allele id.

See Also

popstruct

landscape.write.fdist *writes a landscapes allele frequencies to an FDIST file*

Description

Calculates the allele frequencies in each occupied habitat up to the number specified in popnum. Makes calculations based upon the number of individuals sampled stored in samp

Usage

```
landscape.write.fdist(l, popnum = 20, samp = 24)
```

Arguments

l	landscape object
popnum	max number of populations to sample
samp	max number of individuals to sample per population

Value

NULL

See Also

[landscape.write.foreign,landscape.genepop.output](#)

```
landscape.write.foreign
```

Save a landscape to a file in a foreign format

Description

Save a Rmetasim landscape object to a file in a suite of output formats

Usage

```
rland <- landscape.write.foreign(rland,fn="foreign",numi=24,fmt="GDA")
```

Arguments

rland	the Rmetasim landscape object
fn	the path and name of the file to save the landscape to
numi	number of individuals sampled per population for inclusion in subsequent analyses
fmt	the output format for the landscape: Can take the following values:"arlequin","arlequinhap","biosys","gen

Examples

```
## Needs write access to the current directory, files created!!
exampleland <- landscape.new.example()
landscape.write.foreign(exampleland, fn="exampleland.nex", fmt="GDA")
rm(exampleland)
```

SimulationComponents *Code components to simulate a landscape*

Description

These functions can be used to construct custom simulations of landscapes. Each conducts only a single generations worth of change

Usage

```
rland <- landscape.advance(rland=1)
rland <- landscape.carry(rland=1)
rland <- landscape.extinct(rland=1)
rland <- landscape.reproduce(rland=1)
rland <- landscape.survive(rland=1)
```

Arguments

rland	the Rmetasim landscape object
-------	-------------------------------

Details

`landscape.advance()` merely advances the generation counter and selects the new generations demographic conditions if such conditions can vary. The other functions implement carrying capacity, local extinction, reproduction, and survival/growth, respectively. The function `landscape.simulate()` bundles the functionality of these components into a single function (and executes it slightly faster all within linked C++ code).

See Also

`landscape.simulate`

Index

*Topic **misc**

- is.landscape, 2
- landscape.allelecount, 3
- landscape.allelefreq, 4
- landscape.amova, 4
- landscape.amova.locus, 5
- landscape.clean, 6
- landscape.coalinput, 7
- landscape.democol, 8
- landscape.demography, 9
- landscape.exp.het, 9
- landscape.Fst, 10
- landscape.FWright, 11
- landscape.genepop.output, 12
- landscape.generate.locations, 12
- landscape.locus, 13
- landscape.locus.states, 14
- landscape.locusvec, 15
- landscape.new.epoch, 15
- landscape.new.example, 18
- landscape.new.expression, 18
- landscape.new.floatparam, 19
- landscape.new.individuals, 20
- landscape.new.intparam, 21
- landscape.new.landscape, 22
- landscape.new.local.demo, 23
- landscape.new.locus, 24
- landscape.new.switchparam, 25
- landscape.obs.het, 26
- landscape.ploidy, 26
- landscape.plot.locations, 27
- landscape.pollen.kernel.demo, 28
- landscape.populations, 28
- landscape.sample, 29
- landscape.simulate, 30
- landscape.states, 31
- landscape.test.function, 32
- landscape.theta.h, 32
- landscape.theta.k, 33
- landscape.theta.s, 34
- landscape.to.rtheta, 34
- landscape.write.fdist, 35
- landscape.write.foreign, 36
- SimulationComponents, 36

- is.landscape, 2
- landscape.advance
(SimulationComponents), 36
- landscape.allelecount, 3
- landscape.allelefreq, 4
- landscape.amova, 4
- landscape.amova.locus, 5, 5
- landscape.amova.pairwise, 5, 6
- landscape.carry (SimulationComponents),
36
- landscape.clean, 6
- landscape.coalinput, 7
- landscape.democol, 8
- landscape.demography, 9
- landscape.exp.het, 9
- landscape.extinct
(SimulationComponents), 36
- landscape.Fst, 10
- landscape.FWright, 11
- landscape.genepop.output, 12, 35
- landscape.generate.locations, 12
- landscape.locus, 13
- landscape.locus.states, 14
- landscape.locusvec, 15
- landscape.new.empty
(landscape.new.landscape), 22
- landscape.new.epoch, 15
- landscape.new.example, 18
- landscape.new.expression, 18
- landscape.new.floatparam, 19
- landscape.new.individuals, 20
- landscape.new.intparam, 21
- landscape.new.landscape, 22

- landscape.new.local.demo, [23](#)
- landscape.new.locus, [24](#)
- landscape.new.switchparam, [25](#)
- landscape.obs.het, [26](#)
- landscape.ploidy, [26](#)
- landscape.plot.locations, [27](#)
- landscape.pollen.kernel.demo, [28](#)
- landscape.populations, [28](#)
- landscape.reproduce
 - (SimulationComponents), [36](#)
- landscape.sample, [29](#)
- landscape.simulate, [30](#)
- landscape.states, [31](#)
- landscape.survive
 - (SimulationComponents), [36](#)
- landscape.test.function, [32](#)
- landscape.theta.h, [32](#)
- landscape.theta.k, [33](#)
- landscape.theta.s, [34](#)
- landscape.to.rtheta, [34](#)
- landscape.write.fdist, [35](#)
- landscape.write.foreign, [35](#), [36](#)

SimulationComponents, [36](#)