1 rmetasim landscapes

The main data structure in rmetasim is the 'landscape'. It contains all of the information necessary to specify a simulation rep and contains the exact state of the simulation at any point in time. The implication is that, once built, a landscape can undergo simulation for a period of time and then be:

- analyzed,
- saved,
- or even modified by the user

before moving forward with additional simulation of that landscape.

1.1 Components of a landscape

A landscape is just a 'list()' construct in R. Most of the objects contained in the list are themselves list objects.

It is possible to build one 'by hand' but rmetasim implements a set of convenience functions that make this process much less complicated.

The six main sections of the landscape:

intparam integer parameters describing landscape

switchparam boolean switches turning features on and off

floatparam floating point paramters

demography lists describing the demography of the simulation (complex set of parameters)

loci lists describing the genetic components of the simulation

individuals a matrix that contains all individuals, their demographic states, ages and ids. It also contains the genetic information.

The last three are values can change through the course of the simulation. Especially the last two.

1.1.1 intparam

```
library(rmetasim)
rland <- landscape.new.empty() #creates a skeleton of a landscape
rland <- landscape.new.intparam(rland, h=2, s=2)

names(rland$intparam)
```

```
## [1] "habitats"     "stages"       "locusnum"     "numepochs"
## [5] "currentgen"   "currentepoch" "totalgens"    "numdemos"
## [9] "maxlandsize"
```

These values represent:

habitats the number of different habitats or subpopulations within the landscape

stages the number of stages in the life cycle of the organism; also the size of the S, R, and M matricies for local demography

locusnum the number of different loci currently implemented for the simulation

numepochs the number of different migration scenerios to occur during the simulation

currentgen the current generation the simulation has reached

currentepoch the current epoch the simulation has reached

totoalgens the total number of generations to simulate

numdemos the number of within population demographies

maxlandsize the maxium number of individuals that can exist in the simulation

1.1.2 switchparam

```
rland <- landscape.new.switchparam(rland)
names(rland$switchparam)

## [1] "randepoch"   "randdemo"    "multp"       "densdepdemo"
```

These values represent:

randepoch 1=choose the next epoch randomly using their individual probabilities, 0=choose the next epoch according to their ordering

randdemo 1=assign demographies at random, 0=assign demographies in order

multp 1=multiple paternity, 0=entire families from a single mating

densdepdemo a flag to turn density dependent population regulation on and off

1.1.3 floatparam

```
rland <- landscape.new.floatparam(rland)
names(rland$floatparam)

## [1] "selfing"
```

selfing This value is the selfing rate of the species (between 0 and 1 inclusive) assuming a mixed mating model where selfing occurs S proportion of the time and random mating occurs (1-S) proportion of the time.

1.1.4 demography

```
S <- matrix(c(0.1, 0, 0.5, 0.3), nrow = 2)
R <- matrix(c(0, 1.1, 0, 0), nrow = 2)
M <- matrix(c(0, 0, 0, 1), nrow = 2)
rland <- landscape.new.local.demo(rland,S,R,M)

S <- matrix(c(rep(0,4),
              rep(0,4),
              rep(0,4),
              rep(0,4)), nrow = 4)

R <- matrix(c(0,0,0,1,
              0,0,0,0,
              0,1,0,0,
              0,0,0,0), byrow=T, nrow = 4)

M <- matrix(c(0,0,0,0,
              0,0,0,.1,
              0,0,0,0,
              0,.1,0,0), nrow = 4)

rland <- landscape.new.epoch(rland,S=S,R=R,M=M,extinct=c(.01,0),carry=c(100,200))

names(rland$demography)

## [1] "localdem"  "localdemK" "epochs"

names(rland$demography$localdem[[1]])

## [1] "LocalS" "LocalR" "LocalM"

names(rland$demography$localdemK[[1]])
```

```
## NULL

names(rland$demography$epochs[[1]])

## [1] "RndChooseProb" "StartGen"      "Extinct"       "Carry"
## [5] "Localprob"     "S"             "R"             "M"
```

These are the local S, R, and M matricies for each subpopulation (localdem) and the S, R, and M matrices for the interpopulation movement of individuals (epochs). Notice the subscripts [[1]] after the localdem and epochs. Both localdem and epochs are lists of individual local demographies and epochs respectively. Since both localdem and epochs are lists the subscripts are necessary to view the contents of a single local demography or epoch.

localdem LocalS, LocalR, LocalM: the S, R, and M matricies for a particular local demography during exponential growth. This is the only local demography used if density dependence is turned off.

localdemK LocalS, LocalR, LocalM: the S, R, and M matricies for a particular local demography when at the carrying capacity. Individal demgraphic rates are linearly interpolated between localdem and localdemK based on how close the population size is to carrying capacity.

epochs

RndChooseProb the probability of randomly choosing this epoch if rland$switchparam$randepoch==1

StartGen the generation this epoch begins if rland$switchparam$randepoch==0

Extinct a vector of values given the extinction probability for each subpopulation

Carry a vector of values denoting the maximum individuals for each subpopulation

Localprob a list of probabilities for choosing local demographies if rland$switchparam$randedemo==1

S,R,M The S, R, and M matricies for intersubpopulation movement

```
rland <- landscape.new.locus(rland,type=0,ploidy=2,mutationrate=0.001,
                    transmission=0,numalleles=5)
rland <- landscape.new.locus(rland,type=1,ploidy=1,mutationrate=0.005,
                    numalleles=3,frequencies=c(.2,.2,.6))
rland <- landscape.new.locus(rland,type=2,ploidy=2,mutationrate=0.007,
                    transmission=0,numalleles=6,allelesize=75)
```

```
names(rland$loci[[1]])

## [1] "type"    "ploidy"  "trans"   "rate"    "alleles"

names(rland$loci[[1]]$alleles[[1]])

## [1] "aindex" "birth"  "prop"   "state"
```

loci Both loci and alleles are lists with elements of the structure shown above.

locus **type** The three allele types are 0=Infinite Allele mutation model/Integer state, 1=Strict stepwise mutation model/Integer state, 2=DNA base substitution/variable length sequence state.

ploidy This can be 1 for haploid and 2 for diploid.

trans The mode of inheritance. This can be 0 for bi-parental and 1 for maternal.

rate The per allele (as opposed to per site) mutation rate (range [0 1])

alleles list of alleles

allele **aindex** index number of the allele

birth the generation the allele first appeared

prop frequency of this allele

state an integer for loci type 0 or 1, a string representing sequence of DNA for loci type 2 (only the characters A,T,C,G are allowed)

individuals The last major section is individuals. The structure is merely a large matrix with one individual per row. An example row may look like:

```
rland <- landscape.new.individuals(rland,c(10,15,20,8))
print(rland$individuals[1,])

##  [1] 0 0 0 1 0 0 3 1 3 6 4
```

A (static) example of an individual record with 3 loci (last one haploid) looks like this:.

```
stage notused birthdate ID matID patID loc1a1 loc1a2 loc2a1 loc2a2 loc3a1
```

The first six columns are always present and the last five shown here are a product of the choice of number of loci. The first column contains the individuals subpopulation and lifecycle stage (subpopulation = floor(x/rland$intparam$stages), lifecycle stage = x

mod rland$intparam$stages). The second column is currently unused, always 0. The third column contains the generation in which the individual was born or created. The next three contain numerical ids for the individual, its mother and its father. After the first six columns the indivduals genetic code begins. The loci are shown in order with 2 columns for diploid loci and 1 column for haploid loci. The value of these columns represent the allele index of the allele the individual carries.

1.2 Creating a landscape

The section above introduces the different helper functions that can be used to create landscapes. Help for any of them can be obtained in the classic R fashion: `?landscape.new.intparams` will provide help for the intparam constructor. Typing `landscape.new.example` without the parentheses will show the code for producing an entire landscape.

Here's a complete example of building a landscape with two populations and two stages in each population

```
rland <- landscape.new.empty()
rland <- landscape.new.intparam(rland, h = 2, s = 2)
rland <- landscape.new.switchparam(rland, mp = 0)
rland <- landscape.new.floatparam(rland)

S <- matrix(c(0, 0, 1, 0), byrow = TRUE, nrow = 2)
R <- matrix(c(0, 1.1, 0, 0), byrow = TRUE, nrow = 2)
M <- matrix(c(0, 0, 0, 1), byrow = TRUE, nrow = 2)

rland <- landscape.new.local.demo(rland, S, R, M)

S <- matrix(rep(0, 16), nrow = 4)
R <- matrix(rep(0, 16), nrow = 4)
M <- matrix(rep(0, 16), nrow = 4)

rland <- landscape.new.epoch(rland, S = S, R = R, M = M,
                             carry = c(1000, 1000))

rland <- landscape.new.locus(rland, type = 0, ploidy = 2,
                             mutationrate = 0.001, transmission = 0, numalleles = 5)
rland <- landscape.new.locus(rland, type = 1, ploidy = 1,
                             mutationrate = 0.005, numalleles = 3, frequencies = c(0.2,
                                                                                   0.2, 0.6))
rland <- landscape.new.locus(rland, type = 2, ploidy = 2,
                             mutationrate = 0.007, transmission = 0, numalleles = 6,
                             allelesize = 75)
rland <- landscape.new.individuals(rland, c(50, 0, 50, 0))
```

1.2.1 using (not) pipes with magrittr

The magrittr package implements a way to pipe results from one function to another. This is used extensively in dplyr, for example. Since v3.0.0 of rmetasim, you can use pipes to create landscapes (and act on them as well). Here is the same code above using magrittr instead

```
library(magrittr)
#first set up the matrices for local demographies
S <- matrix(c(0, 0, 1, 0), byrow = TRUE, nrow = 2)
R <- matrix(c(0, 1.1, 0, 0), byrow = TRUE, nrow = 2)
M <- matrix(c(0, 0, 0, 1), byrow = TRUE, nrow = 2)

#and epochs
S.epoch <- matrix(rep(0, 16), nrow = 4)
R.epoch <- matrix(rep(0, 16), nrow = 4)
M.epoch <- matrix(rep(0, 16), nrow = 4)

##now create the landscape
rland <- landscape.new.empty() %>%
    landscape.new.intparam(h=2,s=2) %>%
    landscape.new.switchparam(mp=0) %>%
    landscape.new.floatparam() %>%
    landscape.new.local.demo( S, R, M) %>%
    landscape.new.epoch(S = S.epoch, R = R.epoch, M = M.epoch,
                        carry = c(1000, 1000)) %>%
    landscape.new.locus(type = 0, ploidy = 2,
                        mutationrate = 0.001, transmission = 0, numalleles = 5) %>%
    landscape.new.locus(type = 1, ploidy = 1,
                        mutationrate = 0.005, numalleles = 3, frequencies = c(0.2,
                                                                              0.2, 0.6)) %>%
    landscape.new.locus(type = 2, ploidy = 2,
                        mutationrate = 0.007, transmission = 0, numalleles = 6,
                        allelesize = 75) %>%
    landscape.new.individuals(c(50, 0, 50, 0))
```

2 Run a simulation

You can then use the landscape created in the previous section in a simulation. The easiest approach is to use landscape.simulate(). Here is the result of simulating for 10 time points

(usually thought of as years) and then estimating Φ_{ST} for each locus.

```
rland <- landscape.simulate(rland,10)
landscape.amova(rland)

##          Phi          Phi          Phi
## 0.0003967881 0.0065734855 0.0175187889
```

2.1 With (not) pipes...

Here is the same analysis as the previous section.

```
rland %>% landscape.simulate(10) %>% landscape.amova()

##        Phi        Phi        Phi
## 0.01426222 0.06286929 0.02216500
```