Package 'MGDrivE'

August 19, 2019

Type Package

Title Mosquito Gene Drive Explorer

Version 1.1.0

Maintainer Héctor Manuel Sánchez Castellanos <sanchez.hmsc@berkeley.edu>

URL https://marshalllab.github.io/MGDrivE/,
https://www.marshalllab.com/

BugReports https://github.com/MarshallLab/MGDrivE/issues

Description Provides a model designed to be a reliable testbed where various gene drive interventions for mosquito-borne diseases control. It is being developed to accommodate the use of various mosquito-specific gene drive systems within a population dynamics framework that allows migration of individuals between patches in landscape. Previous work developing the population dynamics can be found in Deredec et al. (2001) <doi:10.1073/pnas.1110717108> and Hancock & Godfray (2007) <doi:10.1186/1475-2875-6-98>,
and extensions to accommodate basic CRISPR homing dynamics in Marshall et al. (2017) <doi:10.1038/s41598-017-02744-7>.

License GPL-3

Encoding UTF-8

Imports R6, Rcpp, Rdpack (>= 0.7), grDevices, graphics, stats, utils, data.table

RdMacros Rdpack

LinkingTo Rcpp

Depends R (>= 2.10)

ByteCompile true

LazyData true

RoxygenNote 6.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Author Héctor Manuel Sánchez Castellanos [aut, cre],
Jared Bennett [aut],
Sean Wu [aut],
John M. Marshall [aut]

Repository CRAN

Date/Publication 2019-08-19 21:00:02 UTC

R topics documented:

accumulate_ADMnew_Patch *Accumulate ADMnew*

Description

Accumulate new ADM males

Usage

```
accumulate_ADMnew_Patch(count)
```

Arguments

count	vector of new ADM males

aggregateFemales *Aggregate Female Output by Genotype*

Description

Aggregate over male mate genotype to convert female matrix output into vector output.

Usage

```
aggregateFemales(readDir, writeDir = NULL, genotypes, remFile = FALSE,
  numCores = 1, verbose = TRUE)
```

Arguments

readDir	Directory to read input from
writeDir	Directory to write output to. Default is readDir
genotypes	Character vector of possible genotypes; found in driveCube$genotypesID
remFile	Boolean flag to remove original (unaggregated) file
numCores	Number of cores when reading/writing. Default is 1.
verbose	Chatty? Default is TRUE

Examples

```
## Not run:
# This example assumes user has already run MGDrivE and generated output.
#  This also assumes that the user has already split output by patch.
# See vignette for complete example.

# set read/write directory
fPath <- "path/to/data/containing/folder"

# Need genotypes from the cube run in the simulation
#  This is dependent on the simulation run
#  Using Mendelian cube for this example
cube <- cubeMendelian()

# no return value from function
# numCores uses data.table::setDTthreads internally
aggregateFemales(readDir= fPath, writeDi = NULL, genotypes = cube$genotypesID,
                 remFile = FALSE, numCores = 1)

## End(Not run)
```

`aggregateOutput` *Aggregate Output Over Landscape*

Description

This function aggregates the output of a run over the entire output, i.e., all of the patches. It writes the output one level above the folder pointed to by readDir, if writeDir is NULL. Output consists of 2 csv files, one for males and one for females, "...M_LandscapeAgg_Run...csv".

Usage

```
aggregateOutput(readDir, writeDir=NULL)
```

Arguments

`readDir`	Directory where output was written to
`writeDir`	Directory to write output to. Default is one level above readDir

Examples

```
## Not run:
# This assumes user has run MGDrivE and output is in fPath.
#  See vignette for examples on how to run MGDrivE

# read/write dirs
fPath <- "folder/containing/output"
oPath <- "folder/to/write/stuff"
```

```
# first, split output by patch and aggregate females by mate genotype
# remember, cube is for example and changes with simulation
cube <- cubeMendelian()

splitOutput(readDir = fPath, writeDir = NULL, remFile = TRUE, numCores = 1)
aggregateFemales(readDir= fPath, writeDi = NULL, genotypes = cube$genotypesID,
                 remFile = FALSE, numCores = 1)

# aggregate mosquitoes over entire landscape
#  no return value
aggregateOutput(readDir = fPath, writeDir = NULL)

## End(Not run)
```

`basicBatchMigration` *Make List of Batch Migration Parameters*

Description

Sets up a list containing the probability of a batch migration, the fractional amount of males/females that migrate, and the weighted probabilities for where to migrate.

Usage

```
basicBatchMigration(batchProbs = 1e-05, sexProbs = c(0.01, 0.01),
  numPatches = 1)
```

Arguments

`batchProbs`	Probability of a batch migration, either 1 number or vector of length equal to the number of patches
`sexProbs`	Population fraction of males and females that migration. Either vector c(M,F) or matrix of 2 columns
`numPatches`	Number of patches in the simulation

Examples

```
# to setup for 3 patches
batchMigration = basicBatchMigration(batchProbs = 1e-5, sexProbs = c(0.1, 0.01), numPatches = 3)
```

basicRepeatedReleases *Make List of Modified Mosquito Releases*

Description

Sets up a release schedule for a single patch, returns a list to be used in oneDay_maleReleases_Patch or oneDay_femaleReleases_Patch.

Usage

```
basicRepeatedReleases(genotypes, releaseStart, releaseEnd, releaseInterval,
  releaseVector, sex = "M")
```

Arguments

genotypes	possible genotypes
releaseStart	day releases start
releaseEnd	day releases end
releaseInterval	interval between releases
releaseVector	named character vector of release composition
sex	character in 'M','F','E'

Examples

```
# to setup for 3 patches but only release in the first with a defined release schedule:

patchReleases = replicate(n = 3,expr = {
  list(maleReleases = NULL,femaleReleases = NULL)
},simplify = FALSE)

patchReleases[[1]]$femaleReleases = basicRepeatedReleases(genotypes = cubeHoming1RA$genotypesID,
                                                          releaseStart = 5,
                                                          releaseEnd = 30,
                                                          releaseInterval = 5,
                                                          releaseVector = c("HH"=100,
                                                                            "Hh"=0,
                                                                            "HR"=0,
                                                                            "hh"=0,
                                                                            "hR"=0,
                                                                            "RR"=0),
                                                          sex = "F")
patchReleases[[1]]$maleReleases = basicRepeatedReleases(genotypes = cubeHoming1RA$genotypesID,
                                                        releaseStart = 50,
                                                        releaseEnd = 60,
                                                        releaseInterval = 1,
                                                        releaseVector = c("HH"=100,
                                                                          "Hh"=0,
```

```
                                    "HR"=0,
                                    "hh"=0,
                                    "hR"=0,
                                    "RR"=0),
 sex = "M")
```

`calcAquaticStagesSurvivalProbability`
Calculate Survival Probability of entire Aquatic Stage Life-cycle

---

Description

Calculate vector of survival probabilities for each stage of aquatic life-cycle.

Usage

```
calcAquaticStagesSurvivalProbability(eggSurvivalProbability,
  larvaSurvivalProbability, pupaSurvivalProbability)
```

Arguments

`eggSurvivalProbability`
see `calcAquaticStageSurvivalProbability`

`larvaSurvivalProbability`
see `calcAquaticStageSurvivalProbability`

`pupaSurvivalProbability`
see `calcAquaticStageSurvivalProbability`

`calcAquaticStageSurvivalProbability`
Calculate Aquatic Stage Survival Probability

Description

Calculate θ_{st}, density-independent survival probability, given by:

$$\theta_{st} = (1 - \mu_{st})^{T_{st}}$$

Usage

```
calcAquaticStageSurvivalProbability(mortalityRate, stageDuration)
```

Arguments

`mortalityRate`	daily mortality probability, μ_{st}
`stageDuration`	duration of aquatic stage, T^{st}

`calcAverageGenerationTime` *Calculate Average Generation Time*

Description

Calculate g, average generation time, given by:

$$g = T_e + T_l + T_p + \frac{1}{\mu_{ad}}$$

Usage

```
calcAverageGenerationTime(stagesDuration, adultMortality)
```

Arguments

`stagesDuration`	vector of lengths of aquatic stages, T_e, T_l, T_p
`adultMortality`	adult mortality rate, μ_{ad}

`calcCos` *Calculate Geodesic Distance - Cosine Method*

Description

This function calculates geodesic distance using the cosine method.

Usage

```
calcCos(latLongs, r = 6378137)
```

Arguments

`latLongs`	Two column matrix of latitudes/longitudes
`r`	Earth radius. Default is WGS-84 radius

Examples

```
# two-column matrix with latitude/longitude, in degrees
latLong = cbind(runif(n = 5, min = 0, max = 90),
                runif(n = 5, min = 0, max = 180))

# cosine distance formula
distMat = calcCos(latLongs = latLong)
```

calcDensityDependentDeathRate *Calculate Density-dependent Larval Mortality*

Description

Calculate α, the strength of density-dependent mortality during the larval stage, given by:

$$\alpha = \left(\frac{1/2 * \beta * \theta_e * Ad_{eq}}{R_m - 1} \right) * \left(\frac{1 - (\theta_l / R_m)}{1 - (\theta_l / R_m)^{1/T_l}} \right)$$

Usage

```
calcDensityDependentDeathRate(fertility, thetaAq, tAq,
  adultPopSizeEquilibrium, populationGrowthRate)
```

Arguments

`fertility`	number of eggs per oviposition for wild-type females, β
`thetaAq`	vector of density-independent survival probabilities of aquatic stages, θ_e, θ_l
`tAq`	vector of lengths of aquatic stages, T_e, T_l, T_p
`adultPopSizeEquilibrium`	adult population size at equilibrium, Ad_{eq}
`populationGrowthRate`	population growth in absence of density-dependent mortality R_m

calcExpKernel *Calculate Exponential Stochastic Matrix*

Description

Given a distance matrix from `calcVinEll`, calculate a stochastic matrix where one step movement probabilities follow an exponential density.

Usage

```
calcExpKernel(distMat, rate)
```

Arguments

`distMat`	distance matrix from `calcVinEll`
`rate`	rate parameter of `Exponential` distribution

Details

The distribution and density functions for the exponential kernel are given below:

$$F(x) = 1 - e^{-\lambda x}$$

$$f(x) = \lambda e^{-\lambda x}$$

where λ is the rate parameter of the exponential distribution.

Examples

```
# setup distance matrix
# two-column matrix with latitude/longitude, in degrees
latLong = cbind(runif(n = 5, min = 0, max = 90),
                runif(n = 5, min = 0, max = 180))

# Vincenty Ellipsoid  distance formula
distMat = calcVinEll(latLongs = latLong)

# calculate exponential distribution over distances
#  rate is just for example
kernMat = calcExpKernel(distMat = distMat, rate = 10)
```

---

calcGammaKernel *Calculate Gamma Stochastic Matrix*

Description

Given a distance matrix from calcVinEll, calculate a stochastic matrix where one step movement probabilities follow a gamma density.

Usage

```
calcGammaKernel(distMat, shape, rate)
```

Arguments

distMat	distance matrix from calcVinEll
shape	shape parameter of GammaDist distribution
rate	rate parameter of GammaDist distribution

Details

The distribution and density functions for the gamma kernel are given below:

$$F(x) = \frac{1}{\Gamma(\alpha)}\gamma(\alpha, \beta x)$$

$$f(x) = \frac{\beta^{\alpha}}{\Gamma(\alpha)}x^{\alpha-1}e^{-\beta x}$$

where $\Gamma(\alpha)$ is the Gamma function, $\gamma(\alpha, \beta x)$ is hte lower incomplete gamma function, and α, β are the shape and rate parameters, respectively.

Examples

```
# setup distance matrix
# two-column matrix with latitude/longitude, in degrees
latLong = cbind(runif(n = 5, min = 0, max = 90),
                runif(n = 5, min = 0, max = 180))

# Vincenty Ellipsoid  distance formula
distMat = calcVinEll(latLongs = latLong)

# calculate gamma distribution over distances
#  shape and rate are just for example
kernMat = calcGammaKernel(distMat = distMat, shape = 1, rate = 1)
```

---

calcHaversine *Calculate Geodesic Distance - Haversine Method*

Description

This function calculates geodesic distance using the Haversine method.

Usage

```
calcHaversine(latLongs, r = 6378137)
```

Arguments

latLongs	Two column matrix of latitudes/longitudes
r	Earth radius. Default is WGS-84 radius

Examples

```
# two-column matrix with latitude/longitude, in degrees
latLong = cbind(runif(n = 5, min = 0, max = 90),
                runif(n = 5, min = 0, max = 180))

# Haversine distance formula
distMat = calcHaversine(latLongs = latLong)
```

`calcHurdleExpKernel` *Calculate Zero-inflated Exponential Stochastic Matrix*

Description

Given a distance matrix from `calcVinEll`, calculate a stochastic matrix where one step movement probabilities follow an zero-inflated exponential density with a point mass at zero. The point mass at zero represents the first stage of a two-stage process, where mosquitoes decide to stay at their current node or leave anywhere. This parameter can be calculated from lifetime probabilities to stay at the current node with the helper function `calcZeroInflation`.

Usage

```
calcHurdleExpKernel(distMat, rate, p0)
```

Arguments

`distMat`	distance matrix from `calcVinEll`
`rate`	rate parameter of `Exponential` distribution
`p0`	point mass at zero

Details

If a mosquito leaves its current node, with probability $1 - p_0$, it then chooses a destination node according to a standard exponential density with rate parameter $rate$.

The distribution and density functions for the zero inflated exponential kernel are given below:

$$F(x) = p_0\theta(x) + (1 - p_0)(1 - e^{-\lambda x})$$

$$f(x) = p_0\delta(x) + (1 - p_0)\lambda e^{-\lambda x}$$

where λ is the rate parameter of the exponential distribution, $\theta(x)$ is the Heaviside step function and $\delta(x)$ is the Dirac delta function.

Examples

```
# setup distance matrix
# two-column matrix with latitude/longitude, in degrees
latLong = cbind(runif(n = 5, min = 0, max = 90),
                runif(n = 5, min = 0, max = 180))

# Vincenty Ellipsoid  distance formula
distMat = calcVinEll(latLongs = latLong)

# calculate hurdle exponential distribution over distances
#  rate and point mass are just for example
kernMat = calcHurdleExpKernel(distMat = distMat, rate = 1/1e6, p0 = 0.1)
```

`calcLarvalPopEquilibrium`

Calculate Equilibrium Larval Population

Description

Equilibrium larval population to sustain population.

Usage

```
calcLarvalPopEquilibrium(alpha, Rm)
```

Arguments

`alpha`	see `calcDensityDependentDeathRate`
`Rm`	see `calcPopulationGrowthRate`

`calcLarvalStageMortalityRate`

Calculate Larval Stage Mortality Rate

Description

Calculate μ_l, the larval mortality, given by

$$\mu_l = 1 - \left(\frac{R_m * \mu_{ad}}{1/2 * \beta * (1 - \mu_m)} \right)^{\frac{1}{T_e + T_l + T_p}}$$

Usage

```
calcLarvalStageMortalityRate(generationPopGrowthRate, adultMortality,
  fertility, aquaticStagesDuration)
```

Arguments

`generationPopGrowthRate`	see `calcPopulationGrowthRate`
`adultMortality`	adult mortality rate, μ_{ad}
`fertility`	number of eggs per oviposition for wild-type females, β
`aquaticStagesDuration`	vector of lengths of aquatic stages, T_e, T_l, T_p

`calcLognormalKernel` *Calculate Lognormal Stochastic Matrix*

Description

Given a distance matrix from `calcVinEll`, calculate a stochastic matrix where one step movement probabilities follow a lognormal density.

Usage

```
calcLognormalKernel(distMat, meanlog, sdlog)
```

Arguments

`distMat`	distance matrix from `calcVinEll`
`meanlog`	log mean of `Lognormal` distribution
`sdlog`	log standard deviation of `Lognormal` distribution

Details

The distribution and density functions for the lognormal kernel are given below:

$$F(x) = \frac{1}{2} + \frac{1}{2}\text{erf}[\frac{\ln x - \mu}{\sqrt{2}\sigma}]$$

$$f(x) = \frac{1}{x\sigma\sqrt{2\pi}}\exp\left(-\frac{(\ln x - \mu)^2}{2\sigma^2}\right)$$

where μ is the mean on the log scale, and σ is the standard deviation on the log scale.

Examples

```
# setup distance matrix
# two-column matrix with latitude/longitude, in degrees
latLong = cbind(runif(n = 5, min = 0, max = 90),
                runif(n = 5, min = 0, max = 180))

# Vincenty Ellipsoid  distance formula
distMat = calcVinEll(latLongs = latLong)

# calculate lognormal distribution over distances
#  mean and standard deviation are just for example
kernMat = calcLognormalKernel(distMat = distMat, meanlog = 100, sdlog = 10)
```

`calcMemoryWindow` *Calculate Memory Window*

Description

Calculates the necessary window of population history required for the model to work

Usage

```
calcMemoryWindow(stagesDuration)
```

Arguments

`stagesDuration`	vector of aquatic stages length (egg/larva/pupa in days)

`calcPopulationGrowthRate` *Calculate Generational Population Growth Rate*

Description

Calculate R_m, population growth in absence of density-dependent mortality, given by:

$$(r_m)^g$$

Usage

```
calcPopulationGrowthRate(dailyPopGrowthRate, averageGenerationTime)
```

Arguments

`dailyPopGrowthRate`	daily population growth rate, r_m
`averageGenerationTime`	see `calcAverageGenerationTime`

`calcQuantiles` *Summary Statistics for Stochastic MGDrivE*

Description

This function reads in all repetitions for each patch and calculates either the mean, quantiles, or both. User chooses the quantiles, up to 4 decimal places, and enters them as a vector. Quantiles are calculated empirically. (order does not matter)

Usage

```
calcQuantiles(readDir, writeDir, mean = TRUE, quantiles = NULL,
  numCores = 1, verbose = TRUE)
```

Arguments

`readDir`	Directory to find repetition folders in
`writeDir`	Directory to write output
`mean`	Boolean, calculate mean or not. Default is TRUE
`quantiles`	Vector of quantiles to calculate. Default is NULL
`numCores`	Number of cores when reading/writing. Default is 1.
`verbose`	Chatty? Default is TRUE

Details

Given the readDir, this function assumes the follow file structure:

- readDir
  - repetition 1
    - patch 1
    - patch 2
    - patch 3
  - repetition 2
    - patch 1
    - patch 2
    - patch 3
  - repetition 3
  - repetition 4
  - ...

Output files are *.csv contain the mean or quantile in the file name, i.e. M/F*Mean*(patchNum).csv and M/F*Quantile*(quantNum)_(patchNum).csv.

Value

Writes output to files in writeDir

Examples

```
## Not run:
# This function assumes network$multRun() has been performed, or several
#  network$oneRun() have been performed and all of the data has been split
#  and aggregated.

# read/write paths
fPath <- "path/to/folder/ofFolders/with/data"
oPath <- "my/path/output"

# here, only calculate mean, no quantiles
#  no return value
calcQuantiles(readDir = fPath, writeDir = oPath, mean = TRUE,
              quantiles = NULL, numCores = 1)

# here, calculate 2.5% and 97.5% quantiles
calcQuantiles(readDir = fPath, writeDir = oPath, mean = FALSE,
              quantiles = c(0.025, 0.975), numCores = 1)

## End(Not run)
```

---

calcVinEll *Calculate Geodesic Distance - Vincenty Ellipsoid Method*

Description

This function calculates geodesic distance using the original Vincenty Ellipsoid method.

Usage

```
calcVinEll(latLongs, a = 6378137, b = 6356752.3142,
  f = 1/298.257223563, eps = 1e-12, iter = 100)
```

Arguments

latLongs	Two column matrix of latitudes/longitudes
a	Equatorial radius of the earth, default is WGS-84 radius
b	Polar radius of the earth, default is WGS-84 radius
f	Flattening or inverse eccentricity, default eccentricity is WGS-84
eps	Convergence criteria
iter	Maximum number of iterations to attempt convergence

Examples

```
# two-column matrix with latitude/longitude, in degrees
latLong = cbind(runif(n = 5, min = 0, max = 90),
                runif(n = 5, min = 0, max = 180))

# Vincenty Ellipsoid  distance formula
distMat = calcVinEll(latLongs = latLong)
```

---

calcVinSph *Calculate Geodesic Distance - Vincenty Sphere Method*

Description

This function calculates geodesic distance using the Vincenty sphere method.

Usage

```
calcVinSph(latLongs, r = 6378137)
```

Arguments

latLongs	Two column matrix of latitudes/longitudes
r	Earth radius. Default is WGS-84 radius

Examples

```
# two-column matrix with latitude/longitude, in degrees
latLong = cbind(runif(n = 5, min = 0, max = 90),
                runif(n = 5, min = 0, max = 180))

# Vincenty Sphere  distance formula
distMat = calcVinSph(latLongs = latLong)
```

---

calcZeroInflation *Calculates the zero-inflation part of a hurdle exponential kernel.*

Description

Given the probability of an adult mosquito to stay in the same patch throughout its whole lifespan, and its mortality, it calculates the height of the pulse-density part of the hurdle kernel.

Usage

```
calcZeroInflation(stayThroughLifespanProbability, adultMortality)
```

Arguments

stayThroughLifespanProbability	Probability of a mosquito to spend its whole lifespan in the same node
adultMortality	Adult mortality rate

Examples

```
# setup distance matrix
# two-column matrix with latitude/longitude, in degrees
latLong = cbind(runif(n = 5, min = 0, max = 90),
                runif(n = 5, min = 0, max = 180))

# Vincenty Ellipsoid  distance formula
distMat = calcVinEll(latLongs = latLong)

# get hurdle height
# Lets assume 80% stay probs and adult mortality of 0.1
hHeight <- calcZeroInflation(stayThroughLifespanProbability = 0.80,
                             adultMortality = 0.1)

# calculate hurdle exponential distribution over distances
kernMat = calcHurdleExpKernel(distMat = distMat, rate = 10, p0 = hHeight)
```

---

close_allConnections_Network

Close all Output Connections

Description

Close `private$conADM` and `private$conAF1`

Usage

```
close_allConnections_Network()
```

---

createNamedPopMatrix

Create a Named Matrix

Description

Create a named matrix of 0s

Usage

```
createNamedPopMatrix(genotypesID)
```

Arguments

genotypesID	character vector of possible genotypes

---

createNamedPopVector *Create a Named Vector*

Description

Create a named vector of 0s

Usage

```
createNamedPopVector(genotypesID)
```

Arguments

genotypesID	character vector of possible genotypes

---

cube2csv *Export a Cube to .csv*

Description

Export a cube as multiple .csv files (one for each genotype; slices of z-axis). This function will create the directory if it doesn't exist. Files are stored as slice_(z-slice)_(genotype).csv

Usage

```
cube2csv(cube, directory, digits = 3)
```

Arguments

cube	A cube object (see MGDrivE-Cube for options)
directory	Directory to write .csv files to
digits	Number of significant digits to retain in .csv output

Examples

```
## Not run:
# output directory
oPath <- "path/to/write/output"

# setup inheritance cube for export, using Mendelian as the example
cube <- cubeMendelian()

# write out
cube2csv(cube = cube, directory = oPath, digits = 3)

## End(Not run)
```

---

cubeHoming1RA *Inheritance Cube: Homing Drive with 1 Resistance Allele*

Description

This function creates an inheritance cube to model a homing gene drive (such as a CRISPR-Cas9 system) that creates 1 type of resistance allele. It assumes no sex-specific inheritance patterns and the construct is on an autosome.

Usage

```
cubeHoming1RA(c = 1, ch = 0, eta = NULL, phi = NULL,
  omega = NULL, xiF = NULL, xiM = NULL, s = NULL)
```

Arguments

c	Cutting rate
ch	Successful homing rate rate
eta	Genotype-specific mating fitness
phi	Genotype-specific sex ratio at emergence
omega	Genotype-specific multiplicative modifier of adult mortality
xiF	Genotype-specific female pupatory success
xiM	Genotype-specific male pupatory success
s	Genotype-specific fractional reduction(increase) in fertility

Value

Named list(inheritance cube, viability mask, genotypes ID, genotypes number, wild-type allele, mating fitness, sex ratio, adult mortality modifier, female pupatory success, male pupatory success, fertility modifier, release genotype)

`cubeHomingDrive` *Inheritance Cube: CRISPR (Clustered Regularly Interspaced Short Palindromic Repeats) with 2 Resistance Alleles and maternal deposition*

Description

This is a sex-specific version of the original cube. It assumes that the construct is on an autosome and there can be different male/female homing rates. It also has maternal deposition, ie, when the male provides a W allele to a female with a H allele, some portion are cut during oogenesis. If the maternal deposition parameters are zero (d* parameters), this is a normal CRISPR drive

Usage

```
cubeHomingDrive(cM = 1, cF = 1, dF = 0, chM = 0, crM = 0,
  chF = 0, crF = 0, dhF = 0, drF = 0, eta = NULL, phi = NULL,
  omega = NULL, xiF = NULL, xiM = NULL, s = NULL)
```

Arguments

`cM`	Male homing rate
`cF`	Female homing rate
`dF`	Female deposition homing rate
`chM`	Male correct homing rate
`crM`	Male resistance generating rate
`chF`	Female correct homing rate
`crF`	Female resistance generating rate
`dhF`	Female correct deposition rate
`drF`	Female resistance deposition rate
`eta`	Genotype-specific mating fitness
`phi`	Genotype-specific sex ratio at emergence
`omega`	Genotype-specific multiplicative modifier of adult mortality
`xiF`	Genotype-specific female pupatory success
`xiM`	Genotype-specific male pupatory success
`s`	Genotype-specific fractional reduction(increase) in fertility

Value

Named list containing the inheritance cube, transition matrix, genotypes, wild-type allele, and all genotype-specific parameters.

`cubeKillerRescue` *Inheritance Cube: Killer-Rescue System*

Description

This function creates an inheritance cube to model a Killer-Rescue system. Killer-Rescue is a 2-locus system: one locus has a toxin and the other locus contains the antidote. The loci are assumed independent and are non-homing.
This drive has 3 alleles at locus 1 and 2 alleles and locus 2:

- Locus 1
  - T: Wild-type allele
  - K: "Killer" toxin allele
  - R: Broken toxin allele
- Locus 2
  - W: Wild-type allele
  - A: Antidote allele

Usage

```
cubeKillerRescue(eR = 0, Keff = 1, Aeff = 1, eta = NULL,
  phi = NULL, omega = NULL, xiF = NULL, xiM = NULL, s = NULL)
```

Arguments

`eR`	Conversion of K allele to R allele, a basal mutation rate
`Keff`	Toxin efficacy
`Aeff`	Antidote efficacy
`eta`	Genotype-specific mating fitness
`phi`	Genotype-specific sex ratio at emergence
`omega`	Genotype-specific multiplicative modifier of adult mortality
`xiF`	Genotype-specific female pupatory success
`xiM`	Genotype-specific male pupatory success
`s`	Genotype-specific fractional reduction(increase) in fertility

Value

Named list containing the inheritance cube, transition matrix, genotypes, wild-type allele, and all genotype-specific parameters.

cubeMEDEA *Inheritance Cube: MEDEA (Maternal Effect Dominant Embryonic Arrest)*

Description

This function creates an inheritance cube to model a MEDEA drive system. This system was first discovered in flour beetles. It biases inheritance by expressing a maternal toxin such that offspring die unless they express a zygotic antidote.
This drive has 3 alleles at 1 locus:

- W: Wild-type allele
- M: MEDEA allele
- R: Resistance allele

Usage

```
cubeMEDEA(rM = 0, rW = 0, Teff = 1, eta = NULL, phi = NULL,
  omega = NULL, xiF = NULL, xiM = NULL, s = NULL)
```

Arguments

rM	Breakdown of MEDEA allele, no homing/toxin/antidote, M -> R conversion
rW	De novo resistance generation, W -> R conversion
Teff	Efficacy of the toxin
eta	Genotype-specific mating fitness
phi	Genotype-specific sex ratio at emergence
omega	Genotype-specific multiplicative modifier of adult mortality
xiF	Genotype-specific female pupatory success
xiM	Genotype-specific male pupatory success
s	Genotype-specific fractional reduction(increase) in fertility

Value

Named list containing the inheritance cube, transition matrix, genotypes, wild-type allele, and all genotype-specific parameters.

cubeMendelian *Inheritance Cube: Mendelian*

Description

This function creates a Mendelian Inheritance Cube. It only handles simple, alphabetic genotypes. The default is 3 alleles at 1 locus, but this can be extended to however many alleles one is interested in, but only at 1 locus.

Usage

```
cubeMendelian(gtype = c("AA", "Aa", "aa"), eta = NULL, phi = NULL,
  omega = NULL, xiF = NULL, xiM = NULL, s = NULL)
```

Arguments

gtype	Vector of genotypes, with the wild-type in the first position
eta	Genotype-specific mating fitness
phi	Genotype-specific sex ratio at emergence
omega	Genotype-specific multiplicative modifier of adult mortality
xiF	Genotype-specific female pupatory success
xiM	Genotype-specific male pupatory success
s	Genotype-specific fractional reduction(increase) in fertility

Value

Named list containing the inheritance cube, transition matrix, genotypes, wild-type allele, and all genotype-specific parameters.

cubeModifiers *Generate and Modify Default Genotype-specific Parameters*

Description

This is an internal function for cubes.

Usage

```
cubeModifiers(gtype, eta = NULL, phi = NULL, omega = NULL,
  xiF = NULL, xiM = NULL, s = NULL)
```

Arguments

gtype	character vector of genotypes
eta	genotype-specific mating fitness
phi	genotype-specific sex ratio at emergence
omega	genotype-specific multiplicative modifier of adult mortality
xiF	genotype-specific female pupatory success
xiM	genotype-specific male pupatory success
s	genotype-specific fractional reduction(increase) in fertility

---

cubeOneLocusTA *Inheritance Cube: 1 Locus Maternal-Toxin/Zygotic-Antidote System*

Description

This function creates a 1 locus maternal-toxin/zygotic-antidote system. This is similar to the construct called UDmel. There is no resistance generation in this model.
This drive has 3 alleles at 1 locus:

- A: Maternal-toxin 1, zygotic-antidote 2
- B: Maternal-toxin 2, zygotic-antidote 1
- W: Wild-type allele

Usage

```
cubeOneLocusTA(TAEfficacy = 1, TBEfficacy = 1, eta = NULL,
  phi = NULL, omega = NULL, xiF = NULL, xiM = NULL, s = NULL)
```

Arguments

TAEfficacy	Maternal toxin A efficacy
TBEfficacy	Maternal toxin B efficacy
eta	Genotype-specific mating fitness
phi	Genotype-specific sex ratio at emergence
omega	Genotype-specific multiplicative modifier of adult mortality
xiF	Genotype-specific female pupatory success
xiM	Genotype-specific male pupatory success
s	Genotype-specific fractional reduction(increase) in fertility

Value

Named list containing the inheritance cube, transition matrix, genotypes, wild-type allele, and all genotype-specific parameters.

`cubeReciprocalTranslocations` *Inheritance Cube: Reciprocal Translocation*

Description

This function creates an inheritance cube to model a reciprocal translocation. This technology was the original form of underdominant system. It involves 2 chromosomes, each with two alleles. This drive has 4 alleles at 2 loci:

- a: Wild-type at locus A
- A: Translocation at locus A
- b: Wile-type at locus B
- B: Translocation at locus B

Usage

```
cubeReciprocalTranslocations(eta = NULL, phi = NULL, omega = NULL,
  xiF = NULL, xiM = NULL, s = NULL)
```

Arguments

`eta`	Genotype-specific mating fitness
`phi`	Genotype-specific sex ratio at emergence
`omega`	Genotype-specific multiplicative modifier of adult mortality
`xiF`	Genotype-specific female pupatory success
`xiM`	Genotype-specific male pupatory success
`s`	Genotype-specific fractional reduction(increase) in fertility

Value

Named list containing the inheritance cube, transition matrix, genotypes, wild-type allele, and all genotype-specific parameters.

cubeRIDL *Inheritance Cube: RIDL (Release of Insects with Dominant Lethality)*

Description

This function creates a RIDL system RIDL (Release of Insects with Dominant Lethality), is a form of SIT. Created by Oxitec, this is based on a positive feedback loop using the toxic tTAV gene, controlled under lab conditions by the TetO promoter. This has 2 alleles at 1 locus

- W: Wild-type allele
- R: OX513 RIDL allele

Usage

```
cubeRIDL(eta = NULL, phi = NULL, omega = NULL, xiF = NULL,
  xiM = NULL, s = NULL)
```

Arguments

eta	Genotype-specific mating fitness
phi	Genotype-specific sex ratio at emergence
omega	Genotype-specific multiplicative modifier of adult mortality
xiF	Genotype-specific female pupatory success
xiM	Genotype-specific male pupatory success
s	Genotype-specific fractional reduction(increase) in fertility

Value

Named list containing the inheritance cube, transition matrix, genotypes, wild-type allele, and all genotype-specific parameters.

cubeTwoLocusTA *Inheritance Cube: 2 Locus Maternal-Toxin/Zygotic-Antidote System*

Description

This function creates a 2 locus maternal-toxin/zygotic-antidote system. This is similar to the construct called UDmel. There is no resistance generation in this model.
This drive has 2 unlinked alleles, 1 allele each at 2 loci:

- A: Maternal-toxin 1, zygotic-antidote 2
- a: Wild-type at locus A
- B: Maternal-toxin 2, zygotic-antidote 1
- b: Wild-type at locus B

Usage

```
cubeTwoLocusTA(TAEfficacy = 1, TBEfficacy = 1, eta = NULL,
  phi = NULL, omega = NULL, xiF = NULL, xiM = NULL, s = NULL)
```

Arguments

TAEfficacy	Maternal toxin A efficacy
TBEfficacy	Maternal toxin B efficacy
eta	Genotype-specific mating fitness
phi	Genotype-specific sex ratio at emergence
omega	Genotype-specific multiplicative modifier of adult mortality
xiF	Genotype-specific female pupatory success
xiM	Genotype-specific male pupatory success
s	Genotype-specific fractional reduction(increase) in fertility

Value

Named list containing the inheritance cube, transition matrix, genotypes, wild-type allele, and all genotype-specific parameters.

---

cubeWolbachia *Inheritance Cube: Wolbachia*

Description

This function creates an inheritance cube to model a Wolbachia infection. Wolbachia is a parasite that can infect mosquitoes. It biases its inheritance through cytoplasmic incompatibility.
This drive has 2 alleles at 1 locus:

- W: has Wolbachia
- w: does not have Wolbachia

Usage

```
cubeWolbachia(eta = NULL, phi = NULL, omega = NULL, xiF = NULL,
  xiM = NULL, s = NULL)
```

Arguments

eta	Genotype-specific mating fitness
phi	Genotype-specific sex ratio at emergence
omega	Genotype-specific multiplicative modifier of adult mortality
xiF	Genotype-specific female pupatory success
xiM	Genotype-specific male pupatory success
s	Genotype-specific fractional reduction(increase) in fertility

Details

Cytoplasmic Incompatibility:

- male W cross female w -> all offspring die (complete penetrance)
- male w cross female W -> all offspring inherit Wolbachia

Value

Named list containing the inheritance cube, transition matrix, genotypes, wild-type allele, and all genotype-specific parameters.

---

eraseDirectory *Erase all files in a directory*

Description

Given a directory path, check it exists and if so delete all its contents.

Usage

```
eraseDirectory(directory, verbose = TRUE)
```

Arguments

directory	directory whose contents will be deleted
verbose	Chatty? Default is TRUE

Examples

```
## Not run:
# Path to directory, can tilde expand
myPath <- "~/path/to/write/output"

# Erase directory
#  No return value
eraseDirectory(directory = myPath)

## End(Not run)
```

`generateReleaseVector` *Make List of Modified Mosquito Releases*

Description

Sets up a release schedule for a single patch, calls `basicRepeatedReleases` internally.

Usage

```
generateReleaseVector(driveCube = driveCube,
  releasesParameters = releasesParameters, sex = "M")
```

Arguments

`driveCube`	gene-drive cube
`releasesParameters`	A list containing the releasesStart, releasesNumber releasesInterval, and releaseProportion named values.
`sex`	character in 'M','F'

Examples

```
# setup a drive cube, using Mendelian as the example
cube <- cubeMendelian()

# setup release parameter list
#  releasesStart is the time of first release
#  releasesNumber is the number of releases
#  releasesInterval is the number of days between releases
#  releaseProportion is the number of mosquitoes released
relParams <- list(releasesStart = 25, releasesNumber = 1,
                  releasesInterval = 0, releaseProportion = 10)

# generate male releases
mRelVec <- generateReleaseVector(driveCube = cube,
                                 releasesParameters = relParams,
                                 sex = "M")

# generate female releases
fRelVec <- generateReleaseVector(driveCube = cube,
                                 releasesParameters = relParams,
                                 sex = "F")
```

getOmega *Solve for Omega (additional genotype-specific mortality)*

Description

Solves for root of equation of geometrically-distributed lifespan for value of omega.

Usage

```
getOmega(mu, lifespanReduction)
```

Arguments

mu	daily mortality probability (discrete-time hazard, called muAd in code)
lifespanReduction	percent reduction in lifespan from average lifespan (target average lifespan will be $\frac{1}{\mu_{Ad}} \times lifespanReduction$)

Examples

```
# reduce lifespan by 10%
#  Example mu is an average for Aedes
newOmega <- getOmega(mu = 0.11, lifespanReduction = 0.90)
```

getOmega_Network *Get omega*

Description

Return genotype-specific multiplicative modifier of adult mortality

Usage

```
getOmega_Network()
```

get_ADMdly_Patch *Get ADMdly*

Description

Return adult male stage delay window population

Usage

```
get_ADMdly_Patch()
```

get_ADMnew_Patch *Get ADMnew*

Description

Return the new ADM females

Usage

```
get_ADMnew_Patch()
```

get_AdPopEQ_Network *Get AdPopEQ*

Description

Return equilibrium adult population

Usage

```
get_AdPopEQ_Network(ix)
```

Arguments

ix	index of patch

get_AF1dly_Patch *Get AF1dly*

Description

Return adult female stage delay window population

Usage

```
get_AF1dly_Patch()
```

get_AF1new_Patch *Get AF1new*

Description

Return the new AF1 females

Usage

```
get_AF1new_Patch()
```

get_alpha_Network *Get alpha*

Description

Return density dependent mortality, see calcDensityDependentDeathRate

Usage

```
get_alpha_Network(ix)
```

Arguments

ix	index of patch

get_batchLocRow_Network *Get Batch Location Distribution*

Description

Return the distribution of location probabilities

Usage

```
get_batchLocRow_Network(ix)
```

Arguments

ix	index of patch

get_batchProbs_Network *Get Batch Migration Probability*

Description

Return the probability of undergoing batch migration each day

Usage

```
get_batchProbs_Network(ix)
```

Arguments

ix	index of patch

get_batchSex_Network *Get Batch Migration Sex*

Description

Return the batch migration size for each sex

Usage

```
get_batchSex_Network(ix, sex)
```

Arguments

ix	index of patch
sex	number, M=1 and F=2

get_beta_Network *Get beta*

Description

Return size of wild-type egg batch

Usage

```
get_beta_Network()
```

`get_conF_Network` *Get conAF1*

Description

Return `connection` where adult female dynamics are written to

Usage

```
get_conF_Network()
```

`get_conM_Network` *Get conADM*

Description

Return `connection` where adult male dynamics are written to

Usage

```
get_conM_Network()
```

`get_dayPopGrowth_Network` *Get dayPopGrowth*

Description

Return daily population growth rate (rm)

Usage

```
get_dayPopGrowth_Network()
```

`get_directory_Network` *Get directory*

Description

Return character string of directory being written to

Usage

```
get_directory_Network()
```

get_drivecubegenotype_Network *Get Element(s) of Drive Cube by Genotype*

Description

Return elements or slices of drive cube. If all NULL return entire cube.

Usage

```
get_drivecubegenotype_Network(fG = NULL, mG = NULL, oG = NULL)
```

Arguments

fG	female genotype
mG	male genotype
oG	offspring genotype

get_drivecubeindex_Network *Get Element(s) of Drive Cube by Index*

Description

Return elements or slices of drive cube. If all NULL return entire cube.

Usage

```
get_drivecubeindex_Network(fG = NULL, mG = NULL, oG = NULL)
```

Arguments

fG	female genotype index
mG	male genotype index
oG	offspring genotype index

get_EGGdly_Patch *Get EGGdly*

Description

Return egg stage delay window population

Usage

```
get_EGGdly_Patch()
```

get_EGG_Patch *Get EGG*

Description

Return egg stage population

Usage

```
get_EGG_Patch()
```

get_eta_Network *Get eta*

Description

Return genotype-specific mating fitness

Usage

```
get_eta_Network()
```

get_femaleMigration_Patch *Get maleMigration*

Description

Return outbound males (nGenotypes X nGenotypes X nPatch array)

Usage

```
get_femaleMigration_Patch()
```

get_F_Patch *Get AF1*

Description

Return adult female stage population

Usage

```
get_F_Patch()
```

get_genotypesID_Network

Get genotypesID

Description

Return character vector of possible genotypes

Usage

```
get_genotypesID_Network()
```

get_genotypesN_Network

Get genotypesN

Description

Return number of possible genotypes

Usage

```
get_genotypesN_Network()
```

get_genPopGrowth_Network

Get genPopGrowth

Description

Return population growth rate, see calcPopulationGrowthRate

Usage

```
get_genPopGrowth_Network()
```

get_g_Network *Get g*

Description

Return average generation time, see calcAverageGenerationTime

Usage

```
get_g_Network()
```

get_LARdly_Patch *Get LARdly*

Description

Return larval stage delay window population

Usage

```
get_LARdly_Patch()
```

get_LAR_Patch *Get LAR*

Description

Return larval stage population

Usage

```
get_LAR_Patch()
```

get_Leq_Network *Get Leq*

Description

Return equilibrium larval population, see calcLarvalPopEquilibrium

Usage

```
get_Leq_Network(ix)
```

Arguments

ix	index of patch

get_maleMigration_Patch *Get maleMigration*

Description

Return outbound males (nGenotypes X nPatch integer matrix)

Usage

```
get_maleMigration_Patch()
```

get_migrationFemaleRow_Network *Get Row of Female Migration Matrix*

Description

Return a matrix object (does not drop dimensions)

Usage

```
get_migrationFemaleRow_Network(ix)
```

Arguments

ix	index of row

get_migrationFemale_Network *Get Female Migration Matrix*

Description

Return a matrix object

Usage

```
get_migrationFemale_Network()
```

get_migrationMaleRow_Network *Get Row of Male Migration Matrix*

Description

Return a matrix object (does not drop dimensions)

Usage

```
get_migrationMaleRow_Network(ix)
```

Arguments

ix	index of row

get_migrationMale_Network *Get Male Migration Matrix*

Description

Return a matrix object

Usage

```
get_migrationMale_Network()
```

get_moveVar_Network *Get moveVar*

Description

Return numeric variance in Dirchlet-Multinomial movement

Usage

```
get_moveVar_Network()
```

get_muAd_Network *Get muAd*

Description

Return adult mortality

Usage

```
get_muAd_Network()
```

get_muAq_Network *Get muAq*

Description

Return larval mortality, see calcLarvalStageMortalityRate

Usage

```
get_muAq_Network()
```

get_M_Patch *Get ADM*

Description

Return adult male stage population

Usage

```
get_M_Patch()
```

get_NetworkPointer_Patch *Get Network Pointer*

Description

Return a reference to the enclosing `Network` object

Usage

```
get_NetworkPointer_Patch()
```

get_nPatch_Network *Get nPatch*

Description

Return number of patches

Usage

```
get_nPatch_Network()
```

get_patches_Network *Get all Patches*

Description

Return a list of `Patch` objects

Usage

```
get_patches_Network()
```

get_patchID_Patch *Get patchID*

Description

Return the ID of this patch

Usage

```
get_patchID_Patch()
```

get_patchReleases_Network *Get Patch Release Schedule*

Description

Return the release schedule for a patch for male or female

Usage

```
get_patchReleases_Network(ix, sex = "M")
```

Arguments

ix	index of patch
sex	character in 'M', 'F'

get_patch_Network *Get Patch*

Description

Return a Patch object

Usage

```
get_patch_Network(ix)
```

Arguments

ix	integer id of patch to return

get_phi_Network *Get phi*

Description

Return genotype-specific sex ratio at emergence

Usage

```
get_phi_Network()
```

get_PUPdly_Patch *Get PUP*

Description

Return pupae stage delay window population

Usage

```
get_PUPdly_Patch()
```

get_PUP_Patch *Get PUP*

Description

Return pupae stage population

Usage

```
get_PUP_Patch()
```

get_releaseType_Network *Get releaseType*

Description

Return genotype of release

Usage

```
get_releaseType_Network()
```

get_simTime_Network *Get simTime*

Description

Return maximum time to run simulation

Usage

```
get_simTime_Network()
```

get_s_Network *Get s*

Description

Return genotype-specific fractional reduction(increase) in fertility

Usage

```
get_s_Network()
```

get_tau_Network *Get Female Viability Mask (tau)*

Description

Get Female Viability Mask (tau)

Usage

```
get_tau_Network(fG = NULL, mG = NULL, oG = NULL)
```

Arguments

fG	Number for which female genotype to get
mG	Number for which male genotype to get
oG	Number for which offspring genotype to get
	Return matrix

get_thetaAq_Network *Get thetaAq*

Description

Return aquatic stage survival probability, see calcAquaticStagesSurvivalProbability and calcAquaticStageSurvival

Usage

```
get_thetaAq_Network(stage)
```

Arguments

stage	character in 'E', 'L', 'P'

get_timeAq_Network *Get timeAq*

Description

Return duration of aquatic stages, see `initStagesDurations`

Usage

```
get_timeAq_Network(stage = NULL)
```

Arguments

`stage`	character in 'E', 'L', 'P'; if `NULL` return total duration

get_tNow_Network *Get tNow*

Description

Return current simulation time

Usage

```
get_tNow_Network()
```

get_wildType_Network *Get wildtype*

Description

Return wild-type genotype

Usage

```
get_wildType_Network()
```

get_windowSize_Network *Get windowSize*

Description

Return memory window size, see `calcMemoryWindow`

Usage

```
get_windowSize_Network()
```

get_xiF_Network *Get xiF*

Description

Return genotype-specific female pupatory success

Usage

```
get_xiF_Network()
```

get_xiM_Network *Get xiM*

Description

Return genotype-specific male pupatory success

Usage

```
get_xiM_Network()
```

ggColUtility *Utility to Imitate ggplot2 Colors*

Description

Sample at equally spaced intervals along the color wheel

Usage

```
ggColUtility(n, alpha = 0.75)
```

Arguments

n	number of colors
alpha	transparency

initPopMatrixArray *Create a population array of matrices*

Description

Creates an array for the population history to be stored. The length of the array is equal to the window required for the model to run (in our specific case it is equal to the sum of aquatic stages lengths).

Usage

```
initPopMatrixArray(genotypesID, memoryWindow)
```

Arguments

genotypesID	character vector of possible genotypes
memoryWindow	integer size of list structure

initPopVectorArray *Create a population array of vectors*

Description

Creates an array for the population history to be stored. The length of the array is equal to the window required for the model to run (in our specific case it is equal to the sum of aquatic stages lengths).

Usage

```
initPopVectorArray(genotypesID, memoryWindow)
```

Arguments

genotypesID	character vector of possible genotypes
memoryWindow	integer size of list structure

initStagesDurations *Initialize Aquatic Stages Durations*

Description

Initialises the vector that holds the duration of each aquatic stage

Usage

```
initStagesDurations(egg = 1, larva = 14, pupa = 1)
```

Arguments

egg	length of egg stage (days)
larva	length of larval stage (days)
pupa	length of pupal stage (days)

kernels *Kernels Parameters*

Description

A named list containing maximum likelihood fitted parameter values from mosquito dispersal estimates.

Usage

```
data(kernels)
```

Format

named list with 5 elements:

lnorm_mean log mean of log-normal density

lnorm_sd log standard deviation of log-normal density

gamma_shape shape parameter of gamma density

gamma_sd rate parameter of gamma density

exp_rate rate parameter of exponential density

MGDrivE *MGDrivE: Mosquito Gene Drive Explorer*

Description

MGDrivE: Mosquito Gene Drive Explorer

Introduction

Recent developments of CRISPR-Cas9 based homing endonuclease gene drive systems for the suppression or replacement of mosquito populations have generated much interest in their use for control of mosquito-borne diseases (such as dengue, malaria, Chikungunya and Zika). This is because genetic control of pathogen transmission may complement or even substitute traditional vector-control interventions, which have had limited success in bringing the spread of these diseases to a halt. Despite excitement for the use of gene drives for mosquito control, current modeling efforts have analyzed only a handful of these new approaches (usually studying just one per framework). Moreover, these models usually consider well-mixed populations with no explicit spatial dynamics. To this end, we are developing MGDrivE (Mosquito Gene DRIVe Explorer), in cooperation with the 'UCI Malaria Elimination Initiative', as a flexible modeling framework to evaluate a variety of drive systems in spatial networks of mosquito populations. This framework provides a reliable testbed to evaluate and optimize the efficacy of gene drive mosquito releases. What separates MGDrivE from other models is the incorporation of mathematical and computational mechanisms to simulate a wide array of inheritance-based technologies within the same, coherent set of equations. We do this

by treating the population dynamics, genetic inheritance operations, and migration between habitats as separate processes coupled together through the use of mathematical tensor operations. This way we can conveniently swap inheritance patterns whilst still making use of the same set of population dynamics equations. This is a crucial advantage of our system, as it allows other research groups to test their ideas without developing new models and without the need to spend time adapting other frameworks to suit their needs.

Brief Description

MGDrivE is based on the idea that we can decouple the genotype inheritance process from the population dynamics equations. This allows the system to be treated and developed in three semi-independent modules that come together to form the system. The way this is done will be described later in this document but a reference diagram is shown here.

Previous Work

The original version of this model was based on work by (Deredec et al. 2011; Hancock and Godfray 2007) and adapted to accommodate CRISPR homing dynamics in a previous publication by our team (Marshall et al. 2017). As it was described, we extended this framework to be able to handle a variable number of genotypes, and migration across spatial scenarios. We did this by adapting the equations to work in a tensor-oriented manner, where each genotype can have different processes affecting their particular strain (death rates, mating fitness, sex-ratio bias, et cetera).

Notation and Conventions

Before beginning the full description of the model we will define some of the conventions we followed for the notation of the written description of the system.

- Overlines are used to denote the dimension of a tensor
- Subscript brackets are used to indicate an element in time. For example: $L_{[t-1]}$ is the larval population at time: $t-1$.
- Parentheses are used to indicate the parameter(s) of a function. For example: $\overline{O(T_e + T_l)}$ represents the function O evaluated with the parameter: $T_e + T_l$
- Matrices follow a 'row-first' indexing order (i: row, j: column)

In the case of one dimensional tensors, each slot represents a genotype of the population. For example, the male population is stored in the following way:

$$\overline{Am} = \begin{pmatrix} g_1 \\ g_2 \\ g_3 \\ \vdots \\ g_n \end{pmatrix}_i$$

All the processes that affect mosquitoes in a genotype-specific way are defined and stored in this way within the framework.

There are two tensors of squared dimensionality in the model: the adult females matrix, and the genotype-specific viability mask. In the case of the former the rows represent the females' genotype,

whilst the columns represent the genotype of the male they mated with:

$$\overline{\overline{Af}} = \begin{pmatrix} g_{11} & g_{12} & g_{13} & \cdots & g_{1n} \\ g_{21} & g_{22} & g_{23} & \cdots & g_{2n} \\ g_{31} & g_{32} & g_{33} & \cdots & g_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{n1} & g_{n2} & g_{n3} & \cdots & g_{nn} \end{pmatrix}_{ij}$$

The genotype-specific viability mask, on the other hand, stores the mothers' genotype in the rows, and the potential eggs' genotype in the columns of the matrix.

References

Deredec A, Godfray HCJ, Burt A (2011). "Requirements for effective malaria control with homing endonuclease genes." *Proceedings of the National Academy of Sciences of the United States of America*, **108**(43), E874–80. ISSN 1091-6490, doi: 10.1073/pnas.1110717108, https://www.pnas.org/content/108/43/E874.

Hancock PA, Godfray HCJ (2007). "Application of the lumped age-class technique to studying the dynamics of malaria-mosquito-human interactions." *Malaria journal*, **6**, 98. ISSN 1475-2875, doi: 10.1186/14752875698, https://malariajournal.biomedcentral.com/articles/10.1186/1475-2875-6-98.

Marshall J, Buchman A, C. HMS, Akbari OS (2017). "Overcoming evolved resistance to population-suppressing homing-based gene drives." *Nature Scientific Reports*, 1–46. ISSN 2045-2322, doi: 10.1038/s41598017027447, https://www.nature.com/articles/s41598-017-02744-7.

`MGDrivE-Cube` *MGDrivE: Inheritance Cube*

Description

To model an arbitrary number of genotypes efficiently in the same mathematical framework we use a 3-dimensional array structure (cube) where each axis represents the following information:

- x: female adult mate genotype
- y: male adult mate genotype
- z: proportion of the offspring that inherits a given genotype (layer)

Details

The cube structure gives us the flexibility to apply tensor operations to the elements within our equations, so that we can calculate the stratified population dynamics rapidly; and within a readable, flexible computational framework. This becomes apparent when we define the equation we use for the computation of eggs laid at any given point in time:

$$\overline{O(T_x)} = \sum_{j=1}^{n} \left(\left((\beta \cdot \overline{s} \cdot \overline{\overline{Af_{[t-T_x]}}}) \cdot \overline{\overline{Ih}} \right) \cdot \Lambda \right)^{\top}_{ij}$$

In this equation, the matrix containing the number of mated adult females ($\overline{\overline{Af}}$) is multiplied element-wise with each one of the layers containing the eggs genotypes proportions expected from this cross ($\overline{\overline{\overline{Ih}}}$). The resulting matrix is then multiplied by a binary 'viability mask' (Λ) that filters out female-parent to offspring genetic combinations that are not viable due to biological impediments (such as cytoplasmic incompatibility). The summation of the transposed resulting matrix returns us the total fraction of eggs resulting from all the male to female genotype crosses ($\overline{O(T_x)}$).

Note: For inheritance operations to be consistent within the framework the summation of each element in the z-axis (this is, the proportions of each one of the offspring's genotypes) must be equal to one.

Drive-specific Cubes

An inheritance cube in an array object that specifies inheritance probabilities (offspring genotype probability) stratified by male and female parent genotypes. MGDrivE provides the following cubes to model different gene drive systems:

- `cubeOneLocusTA`: 1 Locus Maternal-Toxin/Zygotic-Antidote System
- `cubeTwoLocusTA`: 2 Locus Maternal-Toxin/Zygotic-Antidote System
- `cubeHoming1RA`: Homing Drive with 1 Resistance Allele
- `cubeHomingDrive`: CRISPR (Clustered Regularly Interspaced Short Palindromic Repeats) with 2 Resistance Allele
- `cubeKillerRescue`: Killer-Rescue System
- `cubeMEDEA`: MEDEA (Maternal Effect Dominant Embryonic Arrest)
- `cubeReciprocalTranslocations`: Reciprocal Translocation
- `cubeRIDL`: RIDL (Release of Insects with Dominant Lethality)
- `cubeMendelian`: Mendelian
- `cubeWolbachia`: Wolbachia

Functions for Cubes

There are several functions to operate on cube objects.

- `cube2csv`: Export slices of a cube to .csv format

`MGDrivE-Model` *MGDrivE: Model's Mathematical Description*

Description

The original version of this model was based on work by (Deredec et al. 2011; Hancock and Godfray 2007) and adapted to accommodate CRISPR homing dynamics in a previous publication by our team (Marshall et al. 2017). As it was described, we extended this framework to be able to handle a variable number of genotypes, and migration across spatial scenarios. We did this by adapting the equations to work in a tensor-oriented manner, where each genotype can have different processes affecting their particular strain (death rates, mating fitness, sex-ratio bias, et cetera).

Inheritance Cube and Oviposition

To allow the extension of the framework to an arbitrary number of genotypes we decided to transform traditional inheritance matrices into inheritance cubes where each of the axis represents the following information:

- x: female adult mate genotype
- y: male adult mate genotype
- z: proportion of the offspring that inherits a given genotype (slice)

The 'cube' structure gives us the flexibility to apply tensor operations to the elements within our equations, so that we can calculate the stratified population dynamics rapidly; and within a readable, flexible computational framework. This becomes apparent when we define the equation we use for the computation of eggs laid at any given point in time:

$$\overline{O(T_x)} = \sum_{j=1}^{n} \left(\left((\beta * \overline{s} * \overline{\overline{Af_{[t-T_x]}}}) * \overline{\overline{\overline{Ih}}} \right) * \Lambda \right)^{\top}_{ij}$$

In this equation, the matrix containing the number of mated adult females ($\overline{\overline{Af}}$) is multiplied element-wise with each one of the slices containing the eggs genotypes proportions expected from this cross ($\overline{\overline{\overline{Ih}}}$). The resulting matrix is then multiplied by a binary 'viability mask' (Λ) that filters out female-parent to offspring genetic combinations that are not viable due to biological impediments (such as cytoplasmic incompatibility). The summation of the transposed resulting matrix returns us the total fraction of eggs resulting from all the male to female genotype crosses ($\overline{O(T_x)}$).

Note: For inheritance operations to be consistent within the framework the summation of each element in the 'z' axis (this is, the proportions of each one of the offspring's genotypes) must be equal to one.

Population Dynamics

During the three aquatic stages, a density-independent mortality process takes place:

$$\theta_{st} = (1 - \mu_{st})^{T_{st}}$$

Along with a density dependent process dependent on the number of larvae in the environment:

$$F(L[t]) = \left(\frac{\alpha}{\alpha + \sum \overline{L[t]}} \right)^{1/T_l}$$

where α represents the strength of the density-dependent process. This parameter is calculated with:

$$\alpha = \left(\frac{1/2 * \beta * \theta_e * Ad_{eq}}{R_m - 1} \right) * \left(\frac{1 - (\theta_l / R_m)}{1 - (\theta_l / R_m)^{1/T_l}} \right)$$

in which β is the species' fertility in the absence of gene-drives, Ad_{eq} is the adult mosquito population equilibrium size, and R_m is the population growth in the absence of density-dependent mortality. This population growth is calculated with the average generation time (g), the adult mortality rate (μ_{ad}), and the daily population growth rate (r_m):

$$g = T_e + T_l + T_p + \frac{1}{\mu_{ad}} R_m = (r_m)^g$$

Larval Stages: The computation of the larval stage in the population is crucial to the model because the density dependent processes necessary for equilibrium trajectories to be calculated occur here. This calculation is performed with the following equation:

$$D(\theta_l, T_x) = \begin{cases} \theta^{'}_{l[0]} = \theta_l & i = 0 \\ \theta^{'}_{l[i+1]} = \theta^{'}_{l[i]} * F(\overline{L_{[t-i-T_x]}}) & i \leq T_l \end{cases}$$

In addition to this, we need the larval mortality (μ_l):

$$\mu_l = 1 - \left(\frac{R_m * \mu_{ad}}{1/2 * \beta * (1 - \mu_m)} \right)^{\frac{1}{\overline{T_e + T_l + T_p}}}$$

With these mortality processes, we are now able to calculate the larval population:

$$\overline{L_{[t]}} = \overline{L_{[t-1]}} * (1 - \mu_l) * F(\overline{L_{[t-1]}}) + \overline{O(T_e)} * \theta_e - \overline{O(T_e + T_l)} * \theta_e * D(\theta_l, 0)$$

where the first term accounts for larvae surviving one day to the other; the second therm accounts for the eggs that have hatched within the same period of time; and the last term computes the number of larvae that have transformed into pupae.

Adult Stages: We are ultimately interested in calculating how many adults of each genotype exist at any given point in time. For this, we first calculate the number of eggs that are laid and survive to the adult stages with the equation:

$$\overline{E'} = \overline{O(T_e + T_l + T_p)} \left(\overline{\xi_m} * (\theta_e * \theta_p) * (1 - \mu_{ad}) * D(\theta_l, T_p) \right)$$

With this information we can calculate the current number of male adults in the population by computing the following equation:

$$\overline{Am_{[t]}} = \overline{Am_{[t-1]}} * (1 - \mu_{ad}) * \overline{\omega_m} + (1 - \overline{\phi}) * \overline{E'} + \overline{\nu m_{[t-1]}}$$

in which the first term represents the number of males surviving from one day to the next; the second one, the fraction of males that survive to adulthood ($\overline{E'}$) and emerge as males ($1 - \phi$); the last one is used to add males into the population as part of gene-drive release campaigns.

Female adult populations are calculated in a similar way:

$$\overline{\overline{Af_{[t]}}} = \overline{\overline{Af_{[t-1]}}} * (1 - \mu_{ad}) * \overline{\omega_f} + \left(\overline{\phi} * \overline{E'} + \overline{\nu f_{[t-1]}} \right)^{\top} * \left(\frac{\overline{\eta} * \overline{Am_{[t-1]}}}{\sum \overline{Am_{[t-1]}}} \right)$$

where we first compute the surviving female adults from one day to the next; and then we calculate the mating composition of the female fraction emerging from pupa stage. To do this, we obtain the surviving fraction of eggs that survive to adulthood ($\overline{E'}$) and emerge as females (ϕ), we then add the new females added as a result of gene-drive releases ($\overline{\nu f_{[t-1]}}$). After doing this, we calculate the proportion of males that are allocated to each female genotype, taking into account their respective mating fitnesses ($\overline{\eta}$) so that we can introduce the new adult females into the population pool.

Gene Drive Releases and Effects

As it was briefly mentioned before, we are including the option to release both male and/or female individuals into the populations. Another important thing to emphasize is that we allow flexible releases sizes and schedules. Our model handles releases internally as lists of populations compositions so, it is possible to have releases performed at irregular intervals and with different numbers of mosquito genetic compositions as long as no new genotypes are introduced (which have not been previously defined in the inheritance cube).

$$\overline{\nu} = \left\{ \begin{pmatrix} g_1 \\ g_2 \\ g_3 \\ \vdots \\ g_n \end{pmatrix}_{t=1}, \begin{pmatrix} g_1 \\ g_2 \\ g_3 \\ \vdots \\ g_n \end{pmatrix}_{t=2}, \cdots, \begin{pmatrix} g_1 \\ g_2 \\ g_3 \\ \vdots \\ g_n \end{pmatrix}_{t=x} \right\}$$

So far, however, we have not described the way in which the effects of these gene-drives are included into the mosquito populations dynamics. This is done through the use of various modifiers included in the equations:

- $\overline{\omega}$: Relative increase in mortality (zero being full mortality effects and one no mortality effect)
- $\overline{\phi}$: Relative shift in the sex of the pupating mosquitoes (zero biases the sex ratio towards males, whilst 1 biases the ratio towards females).
- $\overline{\eta}$: Standardized mating fitness (zero being complete fitness ineptitude, and one being regular mating skills).
- $\overline{\beta}$: Fecundity (average number of eggs laid).
- $\overline{\xi}$: Pupation success (zero being full mortality and one full pupation success).

Migration

To simulate migration within our framework we are considering patches (or nodes) of fully-mixed populations in a network structure. This allows us to handle mosquito movement across spatially-distributed populations with a transitions matrix, which is calculated with the tensor outer product of the genotypes populations tensors and the transitions matrix of the network as follows:

$$\overline{Am^i_{(t)}} = \sum \overline{A^j_m} \otimes \overline{\overline{\tau m_{[t-1]}}} \overline{\overline{Af^i_{(t)}}} = \sum \overline{\overline{A^j_f}} \otimes \overline{\overline{\tau f_{[t-1]}}}$$

In these equations the new population of the patch i is calculated by summing the migrating mosquitoes of all the j patches across the network defined by the transitions matrix τ, which stores the mosquito migration probabilities from patch to patch. It is worth noting that the migration probabilities matrices can be different for males and females; and that there's no inherent need for them to be static (the migration probabilities may vary over time to accommodate wind changes due to seasonality).

Parameters

This table compiles all the parameters required to run MGDrivE clustered in six categories:

- Life Stages: These deal with the structure of mosquito population.

- Bionomics: This set of parameters is related to the behavior of the specific mosquito species being modeled.
- Gene Drive: Genotype-specific vectors of parameters that affect how each gene-drive modifies the responses of populations to them.
- Releases: List of vectors that control the release of genetically-modified mosquitoes.
- Population: General mosquito-population parameters that control environmentally-determined variables.
- Network: Related to migration between nodes of population units

Stochasticity

MGDrivE allows stochasticity to be included in the dynamics of various processes; in an effort to simulate processes that affect various stages of mosquitoes lives. In the next section, we will describe all the stochastic processes that can be activated in the program. It should be noted that all of these can be turned on and off independently from one another as required by the researcher.

Mosquito Biology: Oviposition

Stochastic egg laying by female/male pairs is separated into two steps: calculating the number of eggs laid by the females and then distributing laid eggs according to their genotypes. The number of eggs laid follows a Poisson distribution conditioned on the number of female/male pairs and the fertility of each female.

$$Poisson(\lambda = numFemales * Fertility)$$

Multinomial sampling, conditioned on the number of offspring and the relative viability of each genotype, determines the genotypes of the offspring.

$$Multinomial\left(numOffspring, p_1, p_2 \dots p_b\right) = \frac{numOffspring!}{p_1!\, p_2\, \dots p_n} p_1^{n_1} p_2^{n_2} \dots p_n^{n_n}$$

Sex Determination

Sex of the offspring is determined by multinomial sampling. This is conditioned on the number of eggs that live to hatching and a probability of being female, allowing the user to design systems that skew the sex ratio of the offspring through reproductive mechanisms.

$$Multinomial(numHatchingEggs, p_{female}, p_{female})$$

Mating Stochastic mating is determined by multinomial sampling conditioned on the number of males and their fitness. It is assumed that females mate only once in their life, therefore each female will sample from the available males and be done, while the males are free to potentially mate with multiple females. The males' ability to mate is modulated with a fitness term, thereby allowing some genotypes to be less fit than others (as seen often with lab releases).

$$Multinomial(numFemales, p_1 f_1, p_2 f_2, \dots p_n f_n)$$

Hatching

Other Stochastic Processes All remaining stochastic processes (larval survival, hatching , pupating, surviving to adult hood) are determined by multinomial sampling conditioned on factors affecting the current life stage. These factors are determined empirically from mosquito population data.

Migration: Variance of stochastic movement (not used in diffusion model of migration). It affects the concentration of probability in the Dirchlet simplex, small values lead to high variance and large values lead to low variance.

References

Deredec A, Godfray HCJ, Burt A (2011). "Requirements for effective malaria control with homing endonuclease genes." *Proceedings of the National Academy of Sciences of the United States of America*, **108**(43), E874–80. ISSN 1091-6490, doi: 10.1073/pnas.1110717108, https://www.pnas.org/content/108/43/E874.

Hancock PA, Godfray HCJ (2007). "Application of the lumped age-class technique to studying the dynamics of malaria-mosquito-human interactions." *Malaria journal*, **6**, 98. ISSN 1475-2875, doi: 10.1186/14752875698, https://malariajournal.biomedcentral.com/articles/10.1186/1475-2875-6-98.

Marshall J, Buchman A, C. HMS, Akbari OS (2017). "Overcoming evolved resistance to population-suppressing homing-based gene drives." *Nature Scientific Reports*, 1–46. ISSN 2045-2322, doi: 10.1038/s41598017027447, https://www.nature.com/articles/s41598-017-02744-7.

`moveMatAll2` *Movement Matrix: All 2*

Description

A movement matrix for simulation with 3 patches.

Usage

```
data(moveMatAll2)
```

Format

A matrix with 3 rows and 3 columns:

Patches 1 and 3 are sources for patch 2 which is a sink.

`moveMatCascade3` *Movement Matrix: Cascade 3*

Description

A movement matrix for simulation with 3 patches.

Usage

```
data(moveMatCascade3)
```

Format

A matrix with 3 rows and 3 columns:

Mosquitoes in patch 1 have equal probability to stay or move to 2; mosquitoes in patch 2 have equal probability to stay or move to 3; mosquitoes in patch 3 stay there.

---

moveMatDiag *Movement Matrix: Diagonal*

Description

A movement matrix for simulation with 3 patches.

Usage

```
data(moveMatDiag)
```

Format

A matrix with 3 rows and 3 columns:

3 independent patches.

---

moveMatDiagOneCity *Movement Matrix: Diagonal One City*

Description

A movement matrix for simulation with 1 patch.

Usage

```
data(moveMatDiagOneCity)
```

Format

A matrix with 1 rows and 1 columns:

A 1 by 1 matrix with entry 1.

moveMatDie *Movement Matrix: Die*

Description

A movement matrix for simulation with 3 patches.

Usage

```
data(moveMatDie)
```

Format

A matrix with 3 rows and 3 columns:

All entries of matrix are 0 for testing that all mosquitoes will be killed.

moveMatIndependent3 *Movement Matrix: Independent 3*

Description

A movement matrix for simulation with 3 patches.

Usage

```
data(moveMatIndependent3)
```

Format

A matrix with 3 rows and 3 columns:

Mosquitoes in patch 1 stay with probability 0.975, move to patch 2 with probability 0.025, mosquitoes in patch 2 and 3 stay in their patches.

moveMatMixedSpil *Movement Matrix: Mixed Spill*

Description

A movement matrix for simulation with 3 patches.

Usage

```
data(moveMatMixedSpil)
```

Format

A matrix with 3 rows and 3 columns:

Mosquitoes in patch 1 stay with probability 0.999, move to patch 2 with probability 0.001, mosquitoes in patch 2 and 3 stay in their patches.

moveMatTaleOfTwoCities *Movement Matrix: Tale of Two Cities*

Description

A movement matrix for simulation with 2 patches.

Usage

```
data(moveMatTaleOfTwoCities)
```

Format

A matrix with 2 rows and 2 columns:

Mosquitoes do not move between the two patches.

moveMatTriDiagonal *Movement Matrix: Tri-diagonal*

Description

A movement matrix for simulation with 12 patches.

Usage

```
data(moveMatTriDiagonal)
```

Format

A matrix with 12 rows and 12 columns:

Tri-diagonal matrix with approximately 0.985 probability on diagonal and rest of probability mass on k-1 and k+1 off-diagonal elements.

moveMatTriple *Movement Matrix: Triple*

Description

A movement matrix for simulation with 3 patches.

Usage

```
data(moveMatTriple)
```

Format

A matrix with 3 rows and 3 columns:

All entries of matrix are 1 for testing that mosquitoes will be produced.

multRun_Network *Run Simulation*

Description

Run multiple simulations on this network

Usage

```
multRun_Network(conM = NULL, conF = NULL, verbose = TRUE)
```

Arguments

conM	Optional vector of connection to write male population dynamics to
conF	Optional vector of connection to write female population dynamics to
verbose	Chatty? Default is TRUE

Network *Network Class Definition*

Description

A Network class object stores all the information for a simulation on a defined landscape.

Usage

```
Network
```

Format

An R6Class generator object

Constructor

- params: see parameterizeMGDrivE
- driveCube: an inheritance cube
- patchReleases: see basicRepeatedReleases for examples on how to set up release schedules
- migrationMale: a stochastic matrix whose dimensions conform to the number of patches
- migrationFemale: a stochastic matrix whose dimensions conform to the number of patches
- directory: character string of output directory

Methods

- get_moveVar: see get_moveVar_Network
- get_timeAq: see get_timeAq_Network
- get_thetaAq: see get_thetaAq_Network
- get_windowSize: see get_windowSize_Network
- get_beta: see get_beta_Network
- get_muAd: see get_muAd_Network
- get_dayPopGrowth: see get_dayPopGrowth_Network
- get_AdPopEQ: see get_AdPopEQ_Network
- get_g: see get_g_Network
- get_genPopGrowth: see get_genPopGrowth_Network
- get_muAq: see get_muAq_Network
- get_alpha: see get_alpha_Network
- get_Leq_Network: see get_Leq_Network
- get_drivecubegenotype: see get_drivecubegenotype_Network
- get_drivecubeindex: see get_drivecubeindex_Network
- get_tau: see get_tau_Network
- get_genotypesID: see get_genotypesID_Network
- get_genotypesN: see get_genotypesN_Network
- get_wildType: see get_wildType_Network
- get_eta: see get_eta_Network
- get_phi: see get_phi_Network
- getOmega: see getOmega_Network
- get_xiF: see get_xiF_Network
- get_xiM: see get_xiM_Network
- get_s: see get_s_Network
- get_releaseType: see get_releaseType_Network
- get_patch: see get_patch_Network
- get_patches: see get_patches_Network
- get_nPatch: see get_nPatch_Network
- get_directory: see get_directory_Network
- get_simTime: see get_simTime_Network
- get_conADM: see get_conM_Network
- get_conAF1: see get_conF_Network
- close_allConnections: see close_allConnections_Network
- get_tNow: see get_tNow_Network
- get_migrationMale: see get_migrationMale_Network

- get_migrationMaleRow: see get_migrationMaleRow_Network
- set_migrationMale: see set_migrationMale_Network
- get_migrationFemale: see get_migrationFemale_Network
- get_migrationFemaleRow: see get_migrationFemaleRow_Network
- set_migrationFemale: see set_migrationFemale_Network
- get_patchReleases: see get_patchReleases_Network
- oneDay_Migration: see oneDay_Migration_Network
- reset: see reset_Network
- oneRun: see oneRun_Network
- multRun: see multRun_Network
- oneDay: see oneDay_Network

Fields

- params: see parameterizeMGDrivE
- patches: a list of Patch objects
- nPatch: number of patches
- simTime: maximum time of simulation
- driveCube: an inheritance cube
- tNow: current time of simulation (time starts at 2 because time 1 is the initial equilibrium state)
- runID: an identifier for the current simulation run, useful for Monte Carlo simulation
- directory: a character string of where to store output
- conADM: a connection to write male population dynamics out to
- conAF1: a connection to write female population dynamics out to
- migrationMale: a stochastic matrix whose dimensions conform to the number of patches
- migrationFemale: a stochastic matrix whose dimensions conform to the number of patches
- migrationBatch: list of items for batch migration in stochastic sim.
- patchReleases: a list of release schedules for each patch
- verbose: Chatty? Default is TRUE

Examples

```
## Not run:
# There are no simple examples for this, so looking at the vignettes would be
#  most useful.

# Complete manual with examples, but none explored in depth.
vignette("MGDrivE-Examples", package = "MGDrivE")

# One example, explored in great detail. This is probably more helpful.
vignette("MGDrivE-Run", package = "MGDrivE")

## End(Not run)
```

`normalise` *Normalise a Numeric Vector*

Description

Normalise a numeric vector to sum to one

Usage

```
normalise(vector)
```

Arguments

`vector`	numeric vector

`oneDay_admPupating_deterministic_Patch` *Deterministc Adult Male Pupation*

Description

Adult male emergence is calculated based on the number of male pupae multiplied by

$$\left(\overline{\xi_m} * (\theta_e * \theta_p) * (1 - \mu_{ad}) * D(\theta_l, T_p)\right)$$

, where $\overline{\xi_m}$ is the genotype-specific pupation success probability.

Usage

```
oneDay_admPupating_deterministic_Patch()
```

`oneDay_admPupating_stochastic_Patch` *Stochastic Adult Male Pupation*

Description

Pupation from male pupae to adults is sampled from a binomial distribution for each genotype where the probability of pupation is given by

$$\left(\overline{\xi_m} * (\theta_e * \theta_p) * (1 - \mu_{ad}) * D(\theta_l, T_p)\right)$$

, where $\overline{\xi_m}$ is the genotype-specific pupation success probability.

Usage

```
oneDay_admPupating_stochastic_Patch()
```

`oneDay_admSurvival_deterministic_Patch`
Determinstc Adult Male Survival

Description

Daily adult male survival is calculated according to

$$\overline{\overline{Am_{[t-1]}} * (1 - \mu_{ad}) * \overline{\omega_m}}$$

, where μ_{ad} is adult mortality rate and $\overline{\omega_m}$ corresponds to genotype-specific mortality effects.

Usage

```
oneDay_admSurvival_deterministic_Patch()
```

`oneDay_admSurvival_stochastic_Patch`
Stochastic Adult Male Survival

Description

Daily adult male survival is sampled from a binomial distribution where survival probability is given by

$$(1 - \mu_{ad}) * \overline{\omega_m}$$

, where μ_{ad} is adult mortality rate and $\overline{\omega_m}$ corresponds to genotype-specific mortality effects.

Usage

```
oneDay_admSurvival_stochastic_Patch()
```

`oneDay_af1Mating_deterministic_Patch`
Determinstc Mating

Description

Mating is calculated as the outer product of newly emerging adult females and adult males, modulated by $\overline{\eta}$, genotype-specific male mating fitness.

Usage

```
oneDay_af1Mating_deterministic_Patch()
```

`oneDay_af1Mating_stochastic_Patch`
Stochastic Mating

Description

Mating for each newly emerging adult female genotype is sampled from a multinomial distribution with probabilities equal to the adult male population vector multiplied by $\overline{\eta}$, genotype-specific male mating fitness.

Usage

```
oneDay_af1Mating_stochastic_Patch()
```

`oneDay_af1Pupation_deterministic_Patch`
Determinstc Adult Female Pupation

Description

Adult female emergence is calculated based on the number of female pupae multiplied by

$$\left(\overline{\xi_f} * (\theta_e * \theta_p) * (1 - \mu_{ad}) * D(\theta_l, T_p)\right)$$

, where $\overline{\xi_f}$ is the genotype-specific pupation success probability.

Usage

```
oneDay_af1Pupation_deterministic_Patch()
```

`oneDay_af1Pupation_stochastic_Patch`
Stochastic Adult Female Pupation

Description

Pupation from female pupae to adults is sampled from a binomial distribution for each genotype where the probability of pupation is given by

$$\left(\overline{\xi_f} * (\theta_e * \theta_p) * (1 - \mu_{ad}) * D(\theta_l, T_p)\right)$$

, where $\overline{\xi_f}$ is the genotype-specific pupation success probability.

Usage

```
oneDay_af1Pupation_stochastic_Patch()
```

`oneDay_af1Survival_deterministic_Patch`
Determinstc Adult Female Survival

Description

Daily adult female survival is calculated according to

$$\overline{\overline{Af_{[t-1]}} * (1 - \mu_{ad}) * \overline{\omega_f}}$$

, where μ_{ad} is adult mortality rate and $\overline{\omega_f}$ corresponds to genotype-specific mortality effects.

Usage

```
oneDay_af1Survival_deterministic_Patch()
```

`oneDay_af1Survival_stochastic_Patch`
Stochastic Adult Female Survival

Description

Daily adult female survival is sampled from a binomial distribution where survival probability is given by

$$(1 - \mu_{ad}) * \overline{\omega_f}$$

, where μ_{ad} is adult mortality rate and $\overline{\omega_f}$ corresponds to genotype-specific mortality effects.

Usage

```
oneDay_af1Survival_stochastic_Patch()
```

`oneDay_calcCumulativeLarvalDensityDependentFactor_Patch`
Calculate Cumulative Larval density-dependent Mortality

Description

Calculate

$$F(L[t]) = \left(\frac{\alpha}{\alpha + \sum \overline{L[t]}} \right)^{1/T_l}$$

, the cumulative effect of density-dependence on larval mortality over the entire duration of larval stage.

Usage

```
oneDay_calcCumulativeLarvalDensityDependentFactor_Patch()
```

`oneDay_calcCumulativePupaDensityDependentFactor_Patch`
Calculate Cumulative Pupae density-dependent Mortality

Description

Calculate

$$D(\theta_l, T_P) = \begin{cases} \theta'_{l[0]} = \theta_l & i = 0 \\ \theta'_{l[i+1]} = \theta'_{l[i]} * F(\overline{L_{[t-i-T_P]}}) & i \leq T_P \end{cases}$$

, the cumulative effect of density-dependence on larval mortality over the entire duration of larval and pupal stages.

Usage

```
oneDay_calcCumulativePupaDensityDependentFactor_Patch()
```

`oneDay_calcLarvalDensityDependentFactor_Patch`
Calculate Larval density-dependent Mortality

Description

Calculate

$$D(\theta_l, T_x) = \begin{cases} \theta'_{l[0]} = \theta_l & i = 0 \\ \theta'_{l[i+1]} = \theta'_{l[i]} * F(\overline{L_{[t-i-T_x]}}) & i \leq T_l \end{cases}$$

, the effect of density-dependence on larval mortality for one day.

Usage

```
oneDay_calcLarvalDensityDependentFactor_Patch()
```

`oneDay_eggReleases2adults_Patch`
Release Eggs into Larval Window

Description

Get eggs that were released into larval window t - aquaStageDuration days ago and put them into the oviposit vector

Usage

```
oneDay_eggReleases2adults_Patch()
```

`oneDay_eggReleases2eggs_Patch`
Release Eggs into hatching eggs

Description

This function performs egg releases into hatching eggs. This allows them to under-go density-dependence

Usage

```
oneDay_eggReleases2eggs_Patch()
```

`oneDay_eggsFract2_deterministic_Patch`
Determinstc Larval Pupation

Description

Calculate the number of larvae that will pupate prior to calculating density-dependent effects in `oneDay_calcCumulativeLarvalDensityDependentFactor_Patch`.

Usage

```
oneDay_eggsFract2_deterministic_Patch()
```

`oneDay_eggsFract2_stochastic_Patch`
Stochastic Larval Pupation

Description

Calculate the number of larvae that will pupate prior to calculating density-dependent effects in `oneDay_calcCumulativeLarvalDensityDependentFactor_Patch`.

Usage

```
oneDay_eggsFract2_stochastic_Patch()
```

oneDay_femaleReleases_Patch

Release Female Mosquitoes in a Patch

Description

Based on this patch's release schedule, handle daily releases.

Usage

```
oneDay_femaleReleases_Patch()
```

oneDay_hatchingFract_deterministic_Patch

Determinstc Fraction of Eggs Maturing to Hatch

Description

Calculate the fraction of hatching eggs accounting for delay given by: $\overline{O(T_e)}$

Usage

```
oneDay_hatchingFract_deterministic_Patch()
```

oneDay_hatchingFract_stochastic_Patch

Stochastic Fraction of Eggs Maturing to Hatch

Description

Calculate the fraction of hatching eggs accounting for delay given by: $\overline{O(T_e)}$ Stochasticity is introduced by assuming $\overline{O(T_e)}$ defines the mean of a Poisson distributed number of eggs that can hatch.

Usage

```
oneDay_hatchingFract_stochastic_Patch()
```

`oneDay_initOutput_Patch`
Initialize Output from Focal Patch

Description

Writes output to the text connections specified in the enclosing `Network`

Usage

```
oneDay_initOutput_Patch()
```

`oneDay_larHatching_deterministic_Patch`
Stochastic Egg Hatching to Larval Stage

Description

Calculate the eggs that have survived and hatched during a day given by: $\overline{O(T_e)} * \theta_e$

Usage

```
oneDay_larHatching_deterministic_Patch()
```

`oneDay_larHatching_stochastic_Patch`
Stochastic Egg Hatching to Larval Stage

Description

Calculate the eggs that have survived and hatched during a day given by: $\overline{O(T_e)} * \theta_e$ The number of eggs that survive to hatch follows a binomial distribution.

Usage

```
oneDay_larHatching_stochastic_Patch()
```

`oneDay_larPupating_deterministic_Patch`
Determinstc Larval Pupation

Description

Calculate the number of larvae that have transformed into pupae given by $\overline{O(T_e + T_l)} * \theta_e * D(\theta_l, 0)$

Usage

```
oneDay_larPupating_deterministic_Patch()
```

`oneDay_larPupating_stochastic_Patch`
Stochastic Larval Pupation

Description

Calculate the number of larvae that have transformed into pupae given by $\overline{O(T_e + T_l)} * \theta_e * D(\theta_l, 0)$ This number follows a binomial distribution.

Usage

```
oneDay_larPupating_stochastic_Patch()
```

`oneDay_larSurvival_deterministic_Patch`
Determinstc Larval Survival

Description

Calculate the number of larvae surviving from day to day, given by:

$$\overline{L_{[t-1]}} * (1 - \mu_l) * F(\overline{L_{[t-1]}})$$

Usage

```
oneDay_larSurvival_deterministic_Patch()
```

oneDay_larSurvival_stochastic_Patch
Stochastic Larval Survival

Description

Calculate the number of larvae surviving from day to day, given by:

$$\overline{L_{[t-1]}} * (1 - \mu_l) * F(\overline{L_{[t-1]}})$$

Stochasticity is introduced by assuming $(1 - \mu_l) * F(\overline{L_{[t-1]}})$ defines binomial likelihood of survival for each genotype of larvae.

Usage

```
oneDay_larSurvival_stochastic_Patch()
```

oneDay_maleReleases_Patch
Release Male Mosquitoes in a Patch

Description

Based on this patch's release schedule, handle daily releases.

Usage

```
oneDay_maleReleases_Patch()
```

oneDay_migrationIn_Patch
Inbound Migration

Description

Accumulate all inbound migration to this patch.

Usage

```
oneDay_migrationIn_Patch(maleIn, femaleIn)
```

Arguments

maleIn	vector of inbound migration
femaleIn	matrix of inbound migration

`oneDay_migrationOut_deterministic_Patch`
Deterministic Outbound Migration from a Patch

Description

Determinstc model of outbound migration of `AF1new` females from this patch, fills up the `femaleMigration` array.

Usage

```
oneDay_migrationOut_deterministic_Patch()
```

`oneDay_migrationOut_stochastic_Patch`
Stochastic Outbound Migration

Description

Stochastic model of migration of `AF1new` females from this patch, fills up the `femaleMigration` array. Migration is modeled as a Dirichlet-Multinomial process parameterized by `moveVar` multiplied by the row corresponding to this patch from the stochastic matrix. A Dirichlet distributed random variate is sampled from `rDirichlet` according to that parameter vector and then movement is sampled from `rmultinom`.

Usage

```
oneDay_migrationOut_stochastic_Patch()
```

`oneDay_Migration_Network`
Inter-Patch Migration

Description

Simulate migration between patches. See `MGDrivE-Model`, 'Migration' section for more details on how inter-patch migration is handled.

Usage

```
oneDay_Migration_Network()
```

oneDay_Network *Run a Single Day on a Network*

Description

Runs a single day of simulation on a Network object, handling population dynamics, migration, population update, and output.

Usage

```
oneDay_Network()
```

oneDay_numMaleFemale_deterministic_Patch *Determinstc Sex Ratio*

Description

Calculate the number of males, $(1 - \overline{\phi}) * \overline{E'}$ and females, $\overline{\phi} * \overline{E'}$

Usage

```
oneDay_numMaleFemale_deterministic_Patch()
```

oneDay_numMaleFemale_stochastic_Patch *Stochastic Sex Ratio*

Description

Calculate the number of males, $(1 - \overline{\phi}) * \overline{E'}$ and females, $\overline{\phi} * \overline{E'}$ These counts are follow a binomial distribution.

Usage

```
oneDay_numMaleFemale_stochastic_Patch()
```

`oneDay_ovipositG1_deterministic_Patch`
Determinstc Oviposition

Description

Calculate the number of eggs oviposited by female mosquitoes following:

$$\overline{O(T_x)} = \sum_{j=1}^{n} \left(\left((\beta * \overline{s} * \overline{\overline{Af_{[t-T_x]}}}) * \overline{\overline{Ih}} \right) * \Lambda \right)_{ij}^{\top}$$

Usage

```
oneDay_ovipositG1_deterministic_Patch()
```

`oneDay_ovipositG1_stochastic_Patch`
Stochastic Oviposition

Description

Calculate the number of eggs oviposited by female mosquitoes following:

$$\overline{O(T_x)} = \sum_{j=1}^{n} \left(\left((\beta * \overline{s} * \overline{\overline{Af_{[t-T_x]}}}) * \overline{\overline{Ih}} \right) * \Lambda \right)_{ij}^{\top}$$

The deterministic result for number of eggs is used as the mean of a Poisson-distributed number of actual eggs oviposited.

Usage

```
oneDay_ovipositG1_stochastic_Patch()
```

`oneDay_PopDynamics_Patch`
Daily Population Dynamics for a Patch

Description

Run population dynamics (not including migration) for this patch.

Usage

```
oneDay_PopDynamics_Patch()
```

oneDay_updatePopulation_Patch *Shift Population*

Description

Update larval and adult populations daily at end of time step, calls `shiftAndUpdatePopVector`

Usage

```
oneDay_updatePopulation_Patch()
```

oneDay_writeOutput_Patch *Write Output from Focal Patch*

Description

Writes output to the text connections specified in the enclosing `Network`

Usage

```
oneDay_writeOutput_Patch()
```

oneRun_Network *Run Simulation*

Description

Run a single simulation on this network.

Usage

```
oneRun_Network(conADM = NULL, conAF1 = NULL, verbose = TRUE)
```

Arguments

`conADM`	Optional `connection` to write male population dynamics to
`conAF1`	Optional `connection` to write female population dynamics to
`verbose`	Chatty? Default is TRUE

parameterizeMGDrivE *parameterizeMGDrivE*

Description

Generate parameters for simulation on a Network. Parameters average generation time g, population growth rate R_m, aquatic mortality μ_{Aq}, and aquatic survival θ_{Aq} are shared between patches and calculated by calcAverageGenerationTime, calcPopulationGrowthRate, calcLarvalStageMortalityRate, and calcAquaticStagesSurvivalProbability.
Patch-specific parameters α and L_{eq} are calculated for each patch by calcDensityDependentDeathRate and calcLarvalPopEquilibrium.

Usage

```
parameterizeMGDrivE(runID = 1L, nPatch, simTime, parallel = FALSE,
  moveVar = 1000, tEgg = 1L, tLarva = 14L, tPupa = 1L, beta = 32,
  muAd = 0.123, popGrowth = 1.096, AdPopEQ, LarPopRatio = NULL,
  AdPopRatio_F = NULL, AdPopRatio_M = NULL)
```

Arguments

runID	begin counting runs with this set of parameters from this value
nPatch	number of Patch
simTime	maximum time to run simulation
parallel	append process id (see link[base]{Sys.getpid}) to output files for running in parallel
moveVar	variance of stochastic movement (not used in diffusion model of migration). It affects the concentration of probability in the Dirchlet simplex, small values lead to high variance and large values lead to low variance.
tEgg	length of egg stage
tLarva	length of larval instar stage
tPupa	length of pupal stage
beta	female egg batch size of wild-type
muAd	wild-type daily adult mortality (1/muAd is average wild-type lifespan)
popGrowth	daily population growth rate (used to calculate equilibrium)
AdPopEQ	vector of adult population size at equilibrium
LarPopRatio	may be NULL; if not, gives the wildtype gene frequencies among larval stages at the beginning of simulation
AdPopRatio_F	may be NULL; if not, gives the wildtype gene frequencies among adult females at the beginning of simulation
AdPopRatio_M	may be NULL; if not, gives the wildtype gene frequencies among adult males at the beginning of simulation

Examples

```
# using default parameters for 2 patches
#  using different population sizes for patches
simPars <- parameterizeMGDrivE(nPatch = 2, simTime = 365,
                               AdPopEQ = c(100,200))
```

`Patch` *Patch Class Definition*

Description

A Patch is a single well-mixed population that is the smallest unit of simulation for MGDrivE.

Usage

```
Patch
```

Format

An `R6Class` generator object

Constructor

- patchID: integer ID of this patch
- genotypesID: character vector of genotypes
- simTime: maximum time of simulation
- windowSize: necessary memory window size for model
- EGGt0: initial egg population, L_{eq}
- LARt0: initial larval population
- PUPt0: initial pupae population
- ADMt0: initial adult male population, Ad_{eq}
- AF1t0: initial adult female population, Ad_{eq}
- maleReleases: integer ID of this patch
- femaleReleases: female release schedule for this patch, see `basicRepeatedReleases`
- larvaeReleases: male release schedule for this patch, see `basicRepeatedReleases`

Methods

- get_patchID: see get_patchID_Patch
- get_AF1new: see get_AF1new_Patch
- set_AF1new: see set_AF1new_Patch
- get_ADMnew: see get_ADMnew_Patch
- set_ADMnew: see set_ADMnew_Patch
- accumulate_ADMnew: see accumulate_ADMnew_Patch
- get_EGG: see get_EGG_Patch
- get_LAR: see get_LAR_Patch
- get_PUP: see get_PUP_Patch
- get_ADM: see get_M_Patch
- get_AF1: see get_F_Patch
- get_EGGdly: see get_EGGdly_Patch
- get_LARdly: see get_LARdly_Patch
- get_PUPdly: see get_PUPdly_Patch
- get_ADMdly: see get_ADMdly_Patch
- get_AF1dly: see get_AF1dly_Patch
- get_maleMigration: see get_maleMigration_Patch
- get_femaleMigration: see get_femaleMigration_Patch
- set_NetworkPointer: see set_NetworkPointer_Patch
- get_NetworkPointer: see get_NetworkPointer_Patch
- reset: see reset_Patch
- oneDay_initOutput: see oneDay_initOutput_Patch
- oneDay_writeOutput: see oneDay_writeOutput_Patch
- oneDay_migrationIn: see oneDay_migrationIn_Patch
- oneDay_maleReleases: see oneDay_maleReleases_Patch
- oneDay_femaleReleases: see oneDay_femaleReleases_Patch
- oneDay_PopDynamics: see oneDay_PopDynamics_Patch
- oneDay_updatePopulation: see oneDay_updatePopulation_Patch
- oneDay_calcLarvalDensityDependentFactor: see oneDay_calcLarvalDensityDependentFactor_Patch
- oneDay_calcCumulativeLarvalDensityDependentFactor: see oneDay_calcCumulativeLarvalDensityDependentFac
- oneDay_calcCumulativePupaDensityDependentFactor: see oneDay_calcCumulativePupaDensityDependentFactor
- oneDay_migrationOut: see oneDay_migrationOut_stochastic_Patch or oneDay_migrationOut_deterministic_
- oneDay_ovipositG1: see oneDay_ovipositG1_stochastic_Patch or oneDay_ovipositG1_deterministic_Patch
- oneDay_larSurvival: see oneDay_larSurvival_stochastic_Patch or oneDay_larSurvival_deterministic_Patc
- oneDay_hatchingFract: see oneDay_hatchingFract_stochastic_Patch or oneDay_hatchingFract_deterministi
- oneDay_larHatching: see oneDay_larHatching_stochastic_Patch or oneDay_larHatching_deterministic_Patc

- oneDay_eggsFract2: see oneDay_eggsFract2_stochastic_Patch or oneDay_eggsFract2_deterministic_Patch
- oneDay_larPupating: see oneDay_larPupating_stochastic_Patch or oneDay_larPupating_deterministic_Patc
- oneDay_numMaleFemale: see oneDay_numMaleFemale_stochastic_Patch or oneDay_numMaleFemale_determinis
- oneDay_admSurvival: see oneDay_admSurvival_stochastic_Patch or oneDay_admSurvival_deterministic_Pat
- oneDay_admPupating: see oneDay_admPupating_stochastic_Patch or oneDay_admPupating_deterministic_Pat
- oneDay_af1Survival: see oneDay_af1Survival_stochastic_Patch or oneDay_af1Survival_deterministic_Patc
- oneDay_af1Pupation: see oneDay_af1Pupation_stochastic_Patch or oneDay_af1Pupation_deterministic_Patc
- oneDay_af1Mating: see oneDay_af1Mating_stochastic_Patch or oneDay_af1Mating_deterministic_Patch

Fields

- patchID: integer ID of this patch
- EGGt0: vector of initial egg stage population
- LARt0: vector of initial larval stage population
- PUPt0: vector of initial pupae stage population
- ADMt0: vector of initial adult male stage population
- AF1t0: matrix of initial adult female stage population
- EGG: egg stage population
- LAR: larvae stage population
- PUP: pupae stage population
- ADM: adult male stage population
- AF1: adult female stage population
- EGGdly: delay egg stage population
- LARdly: delay larvae stage population
- PUPdly: delay pupae stage population
- ADMdly: delay adult male stage population
- AF1dly: delay adult female stage population
- LARnew: new larval population after difference equations; needed to store population prior to migration exchange
- ADMnew: new adult male population after difference equations; needed to store population prior to migration exchange
- AF1new: new adult female population after difference equations; needed to store population prior to migration exchange
- maleMigration: matrix of outbound migrating males of dimension nGenotypes X nPatch
- femaleMigration: array of outbound migrating females of dimension nGenotypes X nGenotypes X nPatch
- NetworkPointer: a reference to enclosing Network
- ovipositG1: new eggs after oviposition by mated female mosquitoes
- larSurvival: surviving larvae

- hatchingFract: fraction of larvae that hatch
- larPupating: fraction of larvae that undergo pupation
- numMaleFemale: number of male vs. female emerging imago stage adults
- admSurvival: number of surviving adult males
- admPupating: number of pupating imago stage adults that become males
- af1Survival: number of surviving adult females
- af1Pupation: number of pupating imago stage adults that become females
- maleMatrix: row of male migration matrix corresponding to migration from this patch
- femaleMatrix: row of female migration matrix corresponding to migration from this patch
- larDDMortal: larval mortality
- f: density dependent factor in larval mortality

`plotMGDrivEMult` *Plot*

Description

Plots several traces from MGDrivE, assuming each set is another repetition from the same experiment.
Given the readDir, this function assumes the follow file structure:

- readDir
  - repetition 1
    - patch 1
    - patch 2
    - patch 3
  - repetition 2
    - patch 1
    - patch 2
    - patch 3
  - repetition 3
  - repetition 4
  - ...

Usage

```
plotMGDrivEMult(readDir, whichPatches = NULL, totalPop = FALSE,
                        nonZeroGen = FALSE, lwd = 0.75, alpha = 0.75)
```

Arguments

readDir	Directory to find repetition folders in
whichPatches	Vector of patches to plot, must be less than 15. Default is NULL if less than 15 patches
totalPop	Boolean, to plot the total population or not. Default is FALSE
nonZeroGen	Boolean, to plot genotypes that are always zero in simulation
lwd	Double, specify the line width for plotting
alpha	Double, specify the opacity for plotting

Details

This function plots output from one run or one set of runs after being analyzed. Setting totalPop to FALSE keeps it from plotting the total population. NonZeroGen accounts for genotypes that could exist, but are not created in the simulation. Default is FALSE, as this is easier to read on a plot.

Examples

```
## Not run:
# Requires the user to have run MGDrivE, logically stochastic, analyzed
#  the data, and stored it in the directory shown below.
# See vignette for complete example

# Folder where single run is stored
fPath <- "path/to/data/containing/folder"

# plot output to see effect
plotMGDrivEMult(readDir=fPath,totalPop = TRUE,lwd=3.5,alpha=1)

## End(Not run)
```

---

plotMGDrivESingle *Plot*

Description

Plots one run from MGDrivE

Usage

```
plotMGDrivESingle(readDir, whichPatches = NULL, totalPop = FALSE,
                          nonZeroGen = FALSE, lwd = 0.75, alpha = 0.75)
```

Arguments

readDir	Path to file from single-run of MGDrivE or from analysis function
whichPatches	Vector of patches to plot, must be less than 15. Default is NULL if less than 15 patches
totalPop	Boolean, to plot the total population or not.
nonZeroGen	Boolean, to plot genotypes that are always zero in simulation
lwd	Double, specify the line width for plotting
alpha	Double, specify the opacity for plotting

Details

This function plots output from one run or one set of runs after being analyzed. Setting totalPop to FALSE keeps it from plotting the total population. NonZeroGen accounts for genotypes that could exist, but are not created in the simulation. Default is FALSE, as this is easier to read on a plot.

Examples

```
## Not run:
# Requires the user to have run MGDrivE, deterministic or stochastic, analyzed
#  the data, and stored it in the directory shown below.
# See vignette for complete example

# Folder where single run is stored
fPath <- "path/to/data/containing/folder"

# plot output to see effect
plotMGDrivESingle(readDir=fPath,totalPop = TRUE,lwd=3.5,alpha=1)

## End(Not run)
```

---

primePopMatrixArray *Create a primed population array of matrices*

Description

Primes an array for the population history to be stored. The length of the array is equal to the window required for the model to run (in our specific case it is equal to the sum of aquatic stages lengths).

Usage

```
primePopMatrixArray(primingMatrix, memoryWindow)
```

Arguments

primingMatrix	a named matrix population
memoryWindow	integer size of list structure

---

primePopVectorArray *Create a primed population array of vectors*

Description

Primes an array for the population history to be stored. The length of the array is equal to the window required for the model to run (in our specific case it is equal to the sum of aquatic stages lengths).

Usage

```
primePopVectorArray(primingVector, memoryWindow)
```

Arguments

primingVector	a named vector population
memoryWindow	integer size of list structure

---

quantileC *Quantiles Function*

Description

Calculate the given quantiles of a matrix.

Usage

```
quantileC(Trials, Probs)
```

Arguments

Trials	Integer matrix to calculate quantiles over
Probs	Vector of quantiles

Details

This function calculates the given quantiles over the rows of an integer matrix. It uses method 8 of the stat::quantiles() function. It gives the same result, to numerical accuracy, and is designed to handle matrix input. It is only designed to work on integer matrices!

Value

Numeric Matrix

rDirichlet *Dirichlet Distribution*

Description

Make a single draw from a Dirichlet distribution with the shape parameter one. This replaces the MCMCpack rDirichlet function, which was wholly written in R.

Usage

```
rDirichlet(migrationPoint)
```

Arguments

migrationPoint	Vector of weights for draws. Must be positive.

reset_Network *Reset Network*

Description

Reset a Network between runs, useful for Monte Carlo simulation. This calls reset_Patch on each patch and resets tNow = 2 and increments the runID.

Usage

```
reset_Network(verbose = TRUE)
```

Arguments

verbose	Chatty? Default is TRUE

reset_Patch *Reset Patch to Initial Conditions*

Description

Resets a patch to its initial configuration so that a new one does not have to be created and allocated in the network (for Monte Carlo simulation).

Usage

```
reset_Patch(verbose = TRUE)
```

Arguments

verbose	Chatty? Default is TRUE

`retrieveOutput` *Retrieve Output*

Description

Read in output from directory. The resulting object will be a nested list; outermost nesting dimension indexes runID, within runID elements are split by sex and innermost nesting is over patches.

Usage

```
retrieveOutput(readDir, verbose = TRUE)
```

Arguments

`readDir`	directory where output was written to; must not end in path separator
`verbose`	Chatty? Default is TRUE

Value

Nested List

Examples

```
## Not run:
# Example assumes user has run and analyzed MGDrivE.
#  See vignette for examples of how to do that.

# set read directory
fPath <- "path/to/split/aggregated/output"

# read in data as nested lists
dataList <- retrieveOutput(readDir = fPath)

## End(Not run)
```

`setupMGDrivE` *Setup MGDrivE*

Description

Initialize methods in `Patch` to run deterministic or stochastic simulations. This sets internal function definitions so that `oneRun_Network` and `multRun_Network` run either deterministic or stochastic functions.

Usage

```
setupMGDrivE(stochasticityON = FALSE, verbose = TRUE)
```

Arguments

stochasticityON	Enable/disable stochastic simulation. Default is FALSE, implying deterministic simulation
verbose	Chatty? Default is TRUE

Examples

```
# run deterministic MGDrivE
setupMGDrivE(stochasticityON = FALSE)

# run stochastic MGDrivE
setupMGDrivE(stochasticityON = TRUE)
```

---

set_ADMnew_Patch *Set ADMnew*

Description

Set an element in new ADM males

Usage

```
set_ADMnew_Patch(count, genotype_M)
```

Arguments

count	number of mosquitoes
genotype_M	genotype of male

set_AF1new_Patch *Set AF1new*

Description

Set an element in new AF1 females

Usage

```
set_AF1new_Patch(count, genotype_F, genotype_M)
```

Arguments

count	number of mosquitoes
genotype_F	genotype of female (row of matrix)
genotype_M	genotype of male (column of matrix)

set_migrationFemale_Network *Set Female Migration Matrix*

Description

Sets link{private$migrationFemale} field

Usage

```
set_migrationFemale_Network(migrationFemale)
```

Arguments

migrationFemale	matrix object (rows must sum to one)

set_migrationMale_Network
Set Male Migration Matrix

Description

Sets link{private$migrationMale} field

Usage

```
set_migrationMale_Network(migrationMale)
```

Arguments

migrationMale	matrix object (rows must sum to one)

set_NetworkPointer_Patch
Set Network Pointer

Description

Set a reference to the enclosing Network object

Usage

```
set_NetworkPointer_Patch(NetworkPointer)
```

Arguments

NetworkPointer	a Network object

shiftAndUpdatePopVector *Shift a Vector*

Description

Shift a population vector by one day and insert the new population. This was written to remove the dependency "binhf".

Usage

```
shiftAndUpdatePopVector(popVector, newPop)
```

Arguments

popVector	List of population vectors of length(Tegg+Tlarva+Tpupa)
newPop	Vector of length equal to the number of genotypes

splitOutput *Split Output by Patch*

Description

Split output into multiple files by patches.

Usage

```
splitOutput(readDir, writeDir = NULL, remFile = TRUE, numCores = 1,
  verbose = TRUE)
```

Arguments

readDir	Directory where output was written to
writeDir	Directory to write output to. Default is readDir
remFile	Remove original output? Default is TRUE
numCores	How many cores to use when writing output. Default is 1
verbose	Chatty? Default is TRUE

Examples

```
## Not run:
# This example assumes user has already run MGDrivE and generated output.
#  If that's untree, see vignette for complete example
fPath <- "path/to/data/containing/folder"
oPath <- "path/to/write/output"

# split data by patch, keep original files
#  not return value
#  numCores uses data.table::setDTthreads internally
splitOutput(readDir = fPath, writeDir = oPath, remFile = FALSE, numCores = 1)

# Alternatively, remove the original files and write new ones in their place
fPath <- "path/to/data/containing/folder"

splitOutput(readDir = fPath, writeDir = NULL, remFile = TRUE, numCores = 1)

## End(Not run)
```

---

turnStochasticityOnOrOff

Enable or Disable Stochastic Model

Description

Set switches for deterministic or stochastic model

Usage

```
turnStochasticityOnOrOff(on = TRUE)
```

Arguments

on	enable/disable stochastic behaviour

Index