Package ‘MGDrivE’

January 20, 2025

Type Package

Title Mosquito Gene Drive Explorer

Version 1.6.0

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 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

RdMacros Rdpack

LinkingTo Rcpp

Depends R (>= 2.10)

ByteCompile true

LazyData true

RoxygenNote 7.1.1

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 2020-10-05 20:40:02 UTC

Contents

`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 = TRUE,
  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
`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
aggregateFemales(readDir= fPath, writeDir = NULL, genotypes = cube$genotypesID,
                 remFile = TRUE)

## 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
#  landscape aggregation will work if females are not aggregated, but it's slower
cube <- cubeMendelian()

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

# 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. The default weights for migration are equal for all patches. These can be changed after running the function. This is only used in oneDay_Migration_Stochastic_Network.

Usage

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

Arguments

`batchProbs`	Probability of a batch migration, either 1 number or a vector of length equal to the number of patches
`sexProbs`	Population fraction of males and females that migrate. Either a 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_releases_Patch` or `oneDay_eggReleases_Patch`. This function is no longer intended to be used alone, please use the standard interface, `generateReleaseVector`.

Usage

```
basicRepeatedReleases(releaseStart, releaseEnd, releaseInterval, releaseMatrix)
```

Arguments

`releaseStart`	Day releases start
`releaseEnd`	Day releases end
`releaseInterval`	Interval between releases
`releaseMatrix`	Numeric matrix specifying the genotype and release amount

Examples

```
## Not run:
# Setup for 3 patches but only release in the first with a defined release
#  schedule, for the cube cubeHomingDrive:

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

patchReleases[[1]]$femaleReleases = MGDrivE::basicRepeatedReleases(releaseStart = 5,
                                                                  releaseEnd = 30,
                                                                  releaseInterval = 5,
                                                           releaseMatrix = matrix(c(5,100),1,2))

patchReleases[[1]]$maleReleases = MGDrivE::basicRepeatedReleases(releaseStart = 50,
                                                                releaseEnd = 60,
                                                                releaseInterval = 1,
                                                           releaseMatrix = matrix(c(5,100),1,2))

## End(Not run)
```

---

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 the 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)
```

`calcLarvalDist` *Calculate Distribution of Larval Population*

Description

This hidden function calculates the distribution of larvae through time by treating the larval-stage as a discrete-time Markov chain, and solving for the stationary distribution. As the only aquatic population known for initializing MGDrivE is the equilibrium larval population, this acts as an anchor from which to calculate the egg and pupae distributions from (see `set_initialPopulation_Patch`).

Usage

```
calcLarvalDist(mu, t)
```

Arguments

`mu`	Double, death rate
`t`	Integer, stage time

`calcLarvalPopEquilibrium` *Calculate Equilibrium Larval Population*

Description

Equilibrium larval population size to sustain adult 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:

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

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

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)
```

`calcOmega` *Solve for Omega (additional genotype-specific mortality)*

Description

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

Usage

```
calcOmega(mu, lifespanReduction)
```

Arguments

`mu`	Daily mortality probability (discrete-time hazard, called `muAd` in code)
`lifespanReduction`	Target reduced lifespan, between 0 and 1 (target average lifespan will be $\frac{1}{\mu_{Ad}} \times lifespanReduction$)

Examples

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

`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, 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
`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)

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

## 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)
```

---

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)
```

cubeClvR *Inheritance Cube: ClvR (Cleave and Rescue)*

Description

Based on the Cleave-and-Rescue system of Oberhofer, this is a 2-locus Cas9-based toxin-antidote system. The first locus carries the Cas9, gRNAs, and a recoded copy of an essential gene. The second locus is the targeted essential gene. This gene can be completely haplosufficient (`hSuf` = 1) or completely haploinsufficient (`hSuf` = 0). It is assumed that having 2 copies of the gene (be it wild-type at the second locus or recoded at the first) confers complete viability.

Usage

```
cubeClvR(
  cF = 1,
  crF = 0,
  ccF = cF,
  ccrF = crF,
  cM = 1,
  crM = 0,
  ccM = cM,
  ccrM = crM,
  dW = 0,
  drW = 0,
  ddW = dW,
  ddrW = drW,
  hSuf = 1,
  eta = NULL,
  phi = NULL,
  omega = NULL,
  xiF = NULL,
  xiM = NULL,
  s = NULL
)
```

Arguments

`cF`	Female cutting rate, one ClvR allele
`crF`	Female functional resistance rate, one ClvR allele
`ccF`	Female cutting rate, two ClvR alleles
`ccrF`	Female functional resistance rate, two ClvR alleles
`cM`	Male cutting rate, one ClvR allele
`crM`	Male functional resistance rate, one ClvR allele
`ccM`	Male cutting rate, two ClvR alleles
`ccrM`	Male functional resistance rate, two ClvR alleles

`dW`	Female deposition cutting rate
`drW`	Female deposition functional resistance rate
`ddW`	Female deposition (HH) cutting rate
`ddrW`	Female deposition (HH) functional resistance rate
`hSuf`	Haplosufficiency level, default is completely sufficient
`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.

`cubeClvR2` *Inheritance Cube: 2-Locus ClvR (Cleave and Rescue)*

Description

Based on the Cleave-and-Rescue system of Oberhofer, this is a 3-locus Cas9-based toxin-antidote system. The first locus carries the Cas9, the second locus carries the gRNAs, and a recoded copy of an essential gene. The third locus is the targeted essential gene. This gene can be completely haplosufficient (`hSuf` = 1) or completely haploinsufficient (`hSuf` = 0). It is assumed that having 2 copies of the gene (be it wild-type at the second locus or recoded at the first) confers complete viability. Additionally, loci 1 and 2 can be linked, given `crM` and `crF`, imitating the original 2-locus ClvR system. For this construct, the first locus will have 2 alleles, the second will have 2 alleles, and the third will have 3 alleles:

- Locus 1
  - W: Wild-type
  - C: Cas9
- Locus 2
  - W: Wild-type
  - G: gRNAs and recoded essential gene
- Locus 3
  - W: Wild-type
  - R: Functional resistant
  - B: Non-functional resistant

Usage

```
cubeClvR2(
  cF = 1,
  crF = 0,
  ccF = cF,
  ccrF = crF,
  cM = 1,
  crM = 0,
  ccM = cM,
  ccrM = crM,
  dW = 0,
  drW = 0,
  ddW = dW,
  ddrW = drW,
  hSuf = 1,
  crossF = 0.5,
  crossM = 0.5,
  eta = NULL,
  phi = NULL,
  omega = NULL,
  xiF = NULL,
  xiM = NULL,
  s = NULL
)
```

Arguments

`cF`	Female cutting rate, one ClvR allele
`crF`	Female functional resistance rate, one ClvR allele
`ccF`	Female cutting rate, two ClvR alleles
`ccrF`	Female functional resistance rate, two ClvR alleles
`cM`	Male cutting rate, one ClvR allele
`crM`	Male functional resistance rate, one ClvR allele
`ccM`	Male cutting rate, two ClvR alleles
`ccrM`	Male functional resistance rate, two ClvR alleles
`dW`	Female deposition cutting rate
`drW`	Female deposition functional resistance rate
`ddW`	Female deposition (HH) cutting rate
`ddrW`	Female deposition (HH) functional resistance rate
`hSuf`	Haplosufficiency level, default is completely sufficient
`crossF`	Crossover rate in females, 0 is completely linked, 0.5 is unlinked, 1.0 is complete divergence
`crossM`	Crossover rate in males, 0 is completely linked, 0.5 is unlinked, 1.0 is complete divergence

`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

Female deposition is implemented incorrectly. Right now, it is performed on male alleles prior to zygote formation - it should happen post-zygote formation. Since this construct doesn't have HDR, this should be fine.
Additionally, it is assumed that deposition requries loaded Cas9-RNP complexes from the mother, having Cas9 and no maternal gRNA, even in the presence of paternal gRNA, will not result in maternal deposition mediated cleavage.

Copy-number dependent rates are based on Cas9, not gRNA. The assumption is that RNA is easier to produce, and therefore won't limit cleavage by Cas9.

Value

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

`cubeECHACR` *Inheritance Cube: ECHACR*

Description

This function creates an ECHACR construct, it has 5 alleles at the first locus and 4 alleles at the second.

- W: Wild-type
- H: Homing allele
- E: Eraser allele
- R: No-cost resistance allele
- B: Detrimental resistance allele
- cHW: Rate of homing from H, W -> H transition
- cEH: Rate of homing from E, H -> E transition
- cEW: Rate of homing from E, W -> E transition

Usage

```
cubeECHACR(
  cHW = 1,
  cEW = 1,
  cEH = 1,
  chHW = 0,
  crHW = 0,
  ceEW = 0,
  crEW = 0,
  ceEH = 0,
  crEH = 0,
  d1 = 0,
  d2 = 0,
  d3 = 0,
  eta = NULL,
  phi = NULL,
  omega = NULL,
  xiF = NULL,
  xiM = NULL,
  s = NULL
)
```

Arguments

`cHW`	Cutting efficiency of drive allele at locus 1
`cEW`	Cutting efficiency of ECHACR element into W
`cEH`	Cutting efficiency of ECHACR element into H
`chHW`	Homing efficiency of drive allele at locus 1
`crHW`	Resistance allele efficiency of drive allele at locus 1
`ceEW`	Homing efficiency of ECHACR element into W
`crEW`	Resistance allele efficiency of ECHACR element into W
`ceEH`	Homing efficiency of ECHACR element into H
`crEH`	Resistance allele efficiency of ECHACR element into H
`d1`	Background mutation rate from W into R allele
`d2`	Background mutation rate from H into R allele
`d3`	Background mutation rate from E into R allele
`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

This inheritance pattern corresponds to the Active Genetic Neutralizing Elements for Halting or Deleting Gene Drives publication.

Value

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

`cubeECHACRX` *Inheritance Cube: ECHACRX*

Description

This function creates an X-linked ECHACR construct, it has 5 alleles at the first locus and 4 alleles at the second.

- W: Wild-type
- H: Homing allele
- E: Eraser allele
- R: No-cost resistance allele
- B: Detrimental resistance allele
- cHW: Rate of homing from H, W -> H transition
- cEH: Rate of homing from E, H -> E transition
- cEW2: Rate of homing from E, W -> E transition

Usage

```
cubeECHACRX(
  cHW = 1,
  cEHW = 1,
  cEW1 = 1,
  cEW2 = 1,
  cEH = 1,
  chHW = 0,
  crHW = 0,
  chEHW = 0,
  crEHW = 0,
  ceEW1 = 0,
  crEW1 = 0,
  ceEW2 = 0,
  crEW2 = 0,
  ceEH = 0,
  crEH = 0,
  d1 = 0,
  d2 = 0,
  d3 = 0,
  dHW = 0,
  dEH = 0,
  dEW = 0,
  drHW = 0,
  drEH = 0,
  drEW = 0,
  crossF = 0,
  eta = NULL,
  phi = NULL,
  omega = NULL,
  xiF = NULL,
  xiM = NULL,
  s = NULL
)
```

Arguments

`cHW`	Cutting efficiency of drive allele at locus 1
`cEHW`	Cutting efficiency of drive allele, in the presence of ECHACR element, at locus 1
`cEW1`	Cutting efficiency of ECHACR element into W at locus 1
`cEW2`	Cutting efficiency of ECHACR element into W at locus 2
`cEH`	Cutting efficiency of ECHACR element into H
`chHW`	Homing efficiency of drive allele at locus 1
`crHW`	Resistance allele efficiency of drive allele at locus 1
`chEHW`	Homing efficiency of drive allele, in the presence of ECHACR element, at locus 1
`crEHW`	Resistance allele efficiency of drive allele, in the presence of ECHACR element, at locus 1
`ceEW1`	Homing efficiency of ECHACR element into W at locus 1
`crEW1`	Resistance allele efficiency of ECHACR element into W at locus 1
`ceEW2`	Homing efficiency of ECHACR element into W at locus 2
`crEW2`	Resistance allele efficiency of ECHACR element into W at locus 2
`ceEH`	Homing efficiency of ECHACR element into H
`crEH`	Resistance allele efficiency of ECHACR element into H
`d1`	Background mutation rate from W into R allele
`d2`	Background mutation rate from H into R allele
`d3`	Background mutation rate from E into R allele
`dHW`	Female H deposition rate against W
`dEH`	Female E deposition rate against H
`dEW`	Female E deposition rate against W

`drHW`	Female resistance generation rate, from H allele
`drEH`	Female resistance generation rate, from E allele
`drEW`	Female resistance generation rate, from E allele
`crossF`	Female crossover rate. 0 is fully linked, 0.5 is unlinked, 1 is negatively linked
`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

This inheritance pattern corresponds to the Active Genetic Neutralizing Elements for Halting or Deleting Gene Drives publication.

Value

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

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

`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 `cubeHoming1RA`. It assumes that the construct is on an autosome and there can be different male/female homing rates. It also has maternal deposition, i.e., 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, 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, handles assortative mating as well
`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.

cubeSplitDrive	*Inheritance Cube: Split CRISPR drive with 2 Resistance Alleles and male/female specific homing*

Description

This is a sex-specific version of a split CRISPR drive. At one locus is the Cas9, inherited in a Mendelian fashion. At a second, unlinked, locus are the gRNAs. When the two loci occur together, the gRNAs drive, with potential damaged alleles, but the Cas9 remains Mendelian. It is assumed that this is an autosomal drive. This drive corresponds to the confinable gene drive system developed by the Akbari lab.

Usage

```
cubeSplitDrive(
  cM = 1,
  chM = 0,
  crM = 0,
  ccM = cM,
  cchM = chM,
  ccrM = crM,
  cF = 1,
  chF = 0,
  crF = 0,
  ccF = cF,
  cchF = chF,
  ccrF = crF,
  dW = 0,
  dhW = 0,
  drW = 0,
  ddW = dW,
  ddhW = dhW,
  ddrW = drW,
  eta = NULL,
  phi = NULL,
  omega = NULL,
  xiF = NULL,
  xiM = NULL,
  s = NULL
)
```

Arguments

cM	Cutting efficiency in males, one Cas9 allele
chM	Homing efficiency in males, one Cas9 allele
crM	Resistance efficiency in males, one Cas9 allele

`ccM`	Cutting efficiency in males, two Cas9 alleles
`cchM`	Homing efficiency in males, two Cas9 alleles
`ccrM`	Resistance efficiency in males, two Cas9 alleles
`cF`	Cutting efficiency in females, one Cas9 allele
`chF`	Homing efficiency in females, one Cas9 allele
`crF`	Resistance efficiency in females, one Cas9 allele
`ccF`	Cutting efficiency in females, two Cas9 alleles
`cchF`	Homing efficiency in females, two Cas9 alleles
`ccrF`	Resistance efficiency in females, two Cas9 alleles
`dW`	Maternal deposition cutting, one Cas9 allele
`dhW`	Maternal deposition homing, one Cas9 allele
`drW`	Maternal deposition resistance, one Cas9 allele
`ddW`	Maternal deposition cutting, two Cas9 alleles
`ddhW`	Maternal deposition homing, two Cas9 alleles
`ddrW`	Maternal deposition resistance, two Cas9 alleles
`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.

`cubeTGD` *Inheritance Cube: tGD*

Description

The trans-complementing Gene Drive (tGD) is a 1-locus, 2 target site drive. The first target site corresponds to the Cas protein, the second to an effector gene and the gRNAs. There are two sets of gRNAs, because each target site may have different cutting/homing/resistance rates, and each sex can have different rates for all of those things. Additionally, the parent that you receive your Cas from dictates its efficiency. Therefor, this construct has 5 alleles at the first locus and 4 alleles at the second.

- Locus 1
  - W: Wild-type

    - P: Paternal Cas9
    - M: Maternal Cas9
    - R: Resistant allele 1
    - B: Resistant allele 2
- Locus 2
    - W: Wild-type
    - G: gRNAs
    - R: Resistant 1
    - B: Resistant 2

Usage

```
cubeTGD(
  cM1 = 0,
  cM2 = 0,
  cP1 = 0,
  cP2 = 0,
  hM1 = 0,
  hM2 = 0,
  hP1 = 0,
  hP2 = 0,
  rM1 = 0,
  rM2 = 0,
  rP1 = 0,
  rP2 = 0,
  crM = 0,
  crP = 0,
  eta = NULL,
  phi = NULL,
  omega = NULL,
  xiF = NULL,
  xiM = NULL,
  s = NULL
)
```

Arguments

`cM1`	Maternally inherited Cas9 cutting rate at locus 1
`cM2`	Maternally inherited Cas9 cutting rate at locus 2
`cP1`	Paternally inherited Cas9 cutting rate at locus 1
`cP2`	Paternally inherited Cas9 cutting rate at locus 2
`hM1`	Maternally inherited Cas9 homing efficiency at locus 1
`hM2`	Maternally inherited Cas9 homing efficiency at locus 2
`hP1`	Paternally inherited Cas9 homing efficiency at locus 1
`hP2`	Paternally inherited Cas9 homing efficiency at locus 2

`rM1`	Maternally inherited Cas9 resistance efficiency at locus 1
`rM2`	Maternally inherited Cas9 resistance efficiency at locus 2
`rP1`	Paternally inherited Cas9 resistance efficiency at locus 1
`rP2`	Paternally inherited Cas9 resistance efficiency at locus 2
`crM`	Maternal crossover rate, 0 is completely linked, 0.5 is unlinked, 1.0 is complete divergence
`crP`	Paternal crossover rate, 0 is completely linked, 0.5 is unlinked, 1.0 is complete divergence
`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

This drive corresponds to the transcomplementing gene drive developed by the Gantz and Bier lab.

Value

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

`cubeTGDX` *Inheritance Cube: tGDX*

Description

The trans-complementing Gene Drive (tGD) is a 1-locus, 2 target site drive. The first target site corresponds to the Cas protein, the second to an effector gene and the gRNAs. There are two sets of gRNAs, because each target site may have different cutting/homing/resistance rates, and each sex can have different rates for all of those things. Additionally, the parent that you receive your Cas from dictates its efficiency. Therefor, this construct has 6 alleles at the first locus and 5 alleles at the second.

- Locus 1
  - W: Wild-type
  - P: Paternal Cas9
  - M: Maternal Cas9
  - R: Resistant allele 1
  - B: Resistant allele 2
  - Y: Y allele

- Locus 2
  - W: Wild-type
  - G: gRNAs
  - R: Resistant 1
  - B: Resistant 2
  - Y: Y allele

Usage

```
cubeTGDX(
  cM1 = 0,
  cM2 = 0,
  cP1 = 0,
  cP2 = 0,
  hM1 = 0,
  hM2 = 0,
  hP1 = 0,
  hP2 = 0,
  rM1 = 0,
  rM2 = 0,
  rP1 = 0,
  rP2 = 0,
  crM = 0,
  eta = NULL,
  phi = NULL,
  omega = NULL,
  xiF = NULL,
  xiM = NULL,
  s = NULL
)
```

Arguments

`cM1`	Maternally inherited Cas9 cutting rate at locus 1
`cM2`	Maternally inherited Cas9 cutting rate at locus 2
`cP1`	Paternally inherited Cas9 cutting rate at locus 1
`cP2`	Paternally inherited Cas9 cutting rate at locus 2
`hM1`	Maternally inherited Cas9 homing efficiency at locus 1
`hM2`	Maternally inherited Cas9 homing efficiency at locus 2
`hP1`	Paternally inherited Cas9 homing efficiency at locus 1
`hP2`	Paternally inherited Cas9 homing efficiency at locus 2
`rM1`	Maternally inherited Cas9 resistance efficiency at locus 1
`rM2`	Maternally inherited Cas9 resistance efficiency at locus 2
`rP1`	Paternally inherited Cas9 resistance efficiency at locus 1
`rP2`	Paternally inherited Cas9 resistance efficiency at locus 2

`crM`	Maternal crossover rate, 0 is completely linked, 0.5 is unlinked, 1.0 is complete divergence
`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

This drive corresponds to the transcomplementing gene drive developed by the Gantz and Bier lab.

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 that 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, releasesParameters, nameGenotypes = NULL)
```

Arguments

driveCube	Gene-drive cube
releasesParameters	A list containing the releasesStart, releasesNumber releasesInterval, and releaseProportion named values.
nameGenotypes	Optional list to specify different genotypes for egg/male/female releases. This is required for mated female releases. This parameter overrides the default release type.

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)

# generate mated female releases
fRelVec <- generateReleaseVector(driveCube = cube,
                                 releasesParameters = relParams,
                                 nameGenotypes = list(c("AA","AA", 10),
                                                      c("AA","aa", 10)))
```

`get_alpha_Network` *Get alpha*

Description

Return density dependent mortality, see `calcDensityDependentDeathRate`

Usage

```
get_alpha_Network(ix)
```

Arguments

`ix`	Index of patch

`get_beta_Network` *Get beta*

Description

Return size of wild-type egg batch

Usage

```
get_beta_Network()
```

`get_conF_Network` *Get conADF*

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_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_eta_Network` *Get eta*

Description

Get eta

Usage

```
get_eta_Network(fIdx)
```

Arguments

`fIdx`	Index of female genotype to pull Return genotype-specific mating fitness

`get_femalePop_Patch` *Get female Population*

Description

Return females (nGenotypes X nGenotypes matrix)

Usage

```
get_femalePop_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_malePop_Patch` *Get male Population*

Description

Return males (nGenotypes vector)

Usage

```
get_malePop_Patch()
```

`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_nPatch_Network` *Get nPatch*

Description

Return number of patches

Usage

```
get_nPatch_Network()
```

`get_omega_Network` *Get omega*

Description

Return genotype-specific multiplicative modifier of adult mortality

Usage

```
get_omega_Network()
```

`get_patchReleases_Network` *Get Patch Release Schedule*

Description

Return the release schedule for a patch for male or female

Usage

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

Arguments

`patch`	Index of patch
`sex`	Character in 'M', 'F', 'Egg', 'mF'

`get_phi_Network` *Get phi*

Description

Return genotype-specific sex ratio at emergence

Usage

```
get_phi_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_timeAq_Network` *Get timeAq*

Description

Return duration of aquatic stages.

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_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

`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 accomplish 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 male-mating ability ($\overline{\eta}$) 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 male mating ability, on the other hand, stores the females' genotype in the rows, and the male genotypes 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:

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

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
- `cubeClvR`: 1 Locus Cleave and Rescue (ClvR)
- `cubeClvR2`: 2 Locus Cleave and Rescue (ClvR)
- `cubeHoming1RA`: Homing Drive with 1 Resistance Allele
- `cubeHomingDrive`: CRISPR (Clustered Regularly Interspaced Short Palindromic Repeats) with 2 Resistance Allele
- `cubeECHACR`: ECHACR (uh....)
- `cubeECHACRX`: ECHACR, X-Linked
- `cubeKillerRescue`: Killer-Rescue System
- `cubeMEDEA`: MEDEA (Maternal Effect Dominant Embryonic Arrest)
- `cubeReciprocalTranslocations`: Reciprocal Translocation
- `cubeRIDL`: RIDL (Release of Insects with Dominant Lethality)
- `cubeMendelian`: Mendelian
- `cubeSplitDrive`: Split CRISPR drive
- `cubeTGD`: trans-complementing Gene Drive
- `cubeTGDX`: trans-complementing Gene Drive, X-Linked
- `cubeWolbachia`: Wolbachia

Functions for Cubes

We provide one auxiliary function 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 transformed 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:

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

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:

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

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

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

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 term 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:

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

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 ($\overline{\overline{\eta}}$) 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 t hing 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).

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

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).
- $\overline{\overline{\eta}}$: 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).

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(verbose = TRUE)
```

Arguments

`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.

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
- migrationBatch: a list of batch migration parameters. See`basicBatchMigration`
- directory: character string of output directory
- verbose: Chatty? Default is TRUE

Methods

- get_timeAq: see `get_timeAq_Network`
- get_beta: see `get_beta_Network`
- get_muAd: see `get_muAd_Network`
- get_muAq: see `get_muAq_Network`
- get_alpha: see `get_alpha_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_eta: see `get_eta_Network`
- get_phi: see `get_phi_Network`
- get_omega: see `get_omega_Network`
- get_xiF: see `get_xiF_Network`
- get_xiM: see `get_xiM_Network`
- get_s: see `get_s_Network`
- get_nPatch: see `get_nPatch_Network`
- get_conADM: see `get_conM_Network`
- get_conADF: see `get_conF_Network`
- get_tNow: see `get_tNow_Network`
- get_patchReleases: see `get_patchReleases_Network`
- oneDay_Migration: see `oneDay_Migration_Deterministic_Network` or see `oneDay_Migration_Stochastic_Netw`
- reset: see `reset_Network`
- oneDay: see `oneDay_Network`
- oneRun: see `oneRun_Network`
- multRun: see `multRun_Network`

Fields

- parameters: see `parameterizeMGDrivE`
- patches: a list of `Patch` objects
- nPatch: number of patches
- simTime: maximum time of simulation
- sampTime: how often to write output, tNow %% sampTime
- 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
- conADF: 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.
- mMoveMat: holder object for male migration
- fMoveArray: holder object for female migration
- patchReleases: a list of release schedules for each patch

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_adultDeath_deterministic_Patch *Deterministic Adult Survival*

Description

Daily adult survival is calculated according to

$$\overline{\overline{Af_{[t-1]}}} * (1 - \mu_{ad}) * \overline{\omega_{m/f}}$$

, where μ_{ad} corresponds to adult mortality rate and $\overline{\omega_{m/f}}$ corresponds to genotype-specific male/female mortality effects.

Usage

```
oneDay_adultDeath_deterministic_Patch()
```

oneDay_adultDeath_stochastic_Patch *Stochastic Adult Survival*

Description

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

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

. μ_{ad} corresponds to adult mortality rate and $\overline{\omega_m / f}$ corresponds to genotype-specific mortality effects.

Usage

```
oneDay_adultDeath_stochastic_Patch()
```

`oneDay_eggDM_deterministic_Patch` *Deterministic Egg Death and Pupation*

Description

Daily egg survival is calculated according to

$$\overline{E_{[t-1]}} * (1 - \mu_{aq})$$

, where μ_{aq} corresponds to daily non-density-dependent aquatic mortality. Eggs transition into larvae at the end of T_e.
See `parameterizeMGDrivE` for how these parameters are derived.

Usage

```
oneDay_eggDM_deterministic_Patch()
```

`oneDay_eggDM_stochastic_Patch` *Stochastic Egg Death and Pupation*

Description

Daily egg survival is sampled from a binomial distribution, where survival probability is given by $1 - \mu_{aq}$. μ_{aq} corresponds to daily non-density-dependent aquatic mortality.
Eggs transition into larvae at the end of T_e.
See `parameterizeMGDrivE` for how these parameters are derived.

Usage

```
oneDay_eggDM_stochastic_Patch()
```

`oneDay_eggReleases_Patch` *Release Eggs in a Patch*

Description

Based on this patch's release schedule, `generateReleaseVector`, this function handles daily egg releases.

Usage

```
oneDay_eggReleases_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_larvaDM_deterministic_Patch`

Deterministic Larva Death and Pupation

Description

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

$$\overline{L_{[t-1]}} * (1 - \mu_{aq}) * F(L)$$

. F(L), the density dependence is calculated as

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

. See `parameterizeMGDrivE` for how these parameters are derived. Pupation has no parameters, so the final day of larvae naturally enter the pupal state.

Usage

```
oneDay_larvaDM_deterministic_Patch()
```

`oneDay_larvaDM_stochastic_Patch` *Stochastic Larva Death and Pupation*

Description

The daily number of larvae surviving is drawn from a binomial distribution, where survival probability is given by

$$(1-\mu_{aq}) * F(L)$$

. F(L), the density dependence is calculated as

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

. See `parameterizeMGDrivE` for how these parameters are derived. Pupation has no parameters, so the final day of larvae naturally enter the pupal state.

Usage

```
oneDay_larvaDM_stochastic_Patch()
```

`oneDay_mating_deterministic_Patch` *Deterministic Mating*

Description

Mating is calculated as the outer product of newly emerging adult females and all-current adult males, modulated by $\overline{\overline{\eta}}$, the genotype-specific male mating fitness. $\overline{\overline{\eta}}$ corresponds to all female (rows) and male (columns) genotypes, to perform any type of assortative mating.
If there are no adult males, the unmated females experience one day of death, calculated as

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

, and remain unmated until tomorrow.

Usage

```
oneDay_mating_deterministic_Patch()
```

`oneDay_mating_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 $\overline{\overline{\eta}}$, the genotype-specific male mating fitness. $\overline{\overline{\eta}}$ corresponds to all female (rows) and male (columns) genotypes, to perform any type of assortative mating.
If there are no adult males, the unmated females experience one day of death, sampled from a binomial distribution parameterized by

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

, and remain unmated until tomorrow.

Usage

```
oneDay_mating_stochastic_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_Migration_Deterministic_Network` *Deterministic Inter-Patch Migration*

Description

Deterministic model of interpatch migration from each patch. `popFemale`/`popMale` is retrieved from each patch using `get_femalePop_Patch`/`get_malePop_Patch`. Migration location is determined from the supplied matrices, `private$migrationFemale` or `private$migrationMale`. Final migration status is held in `private$fMoveArray` or `private$mMoveMat`.
Batch migration is not used in the deterministic model.

Usage

```
oneDay_Migration_Deterministic_Network()
```

Details

This function handles outbound and inbound migration. See `MGDrivE-Model`, 'Migration' section for more details on how inter-patch migration is handled.

`oneDay_Migration_Stochastic_Network` *Stochastic Inter-Patch Migration*

Description

Stochastic model of interpatch migration from each patch. `popFemale`/`popMale` is retrieved from each patch using `get_femalePop_Patch`/`get_malePop_Patch`. Migration location is determined from the supplied matrices, `private$migrationFemale` or `private$migrationMale`. Migration is modeled as a Multinomial process parameterized by migration location probabilities corresponding to each patch . Movement is sampled from `rmultinom`.
Batch migration begins as a `rbinom` sampled from `private$migrationBatch$batchProbs`.If there is batch migration, the location of migration is sampled uniformly (see `sample`), parameterized by `private$migrationBatch$moveProbs`. The amount of each sex that migrations is sampled from `rbinom`, parameterized by `private$migrationBatch$sexProbs`.

Usage

```
oneDay_Migration_Stochastic_Network()
```

Details

This function handles outbound and inbound migration. See `MGDrivE-Model`, 'Migration' section for more details on how inter-patch migration is handled.

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_oviposit_deterministic_Patch *Deterministic Oviposition*

Description

Calculate the number of eggs oviposited by female mosquitoes following:

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

Usage

```
oneDay_oviposit_deterministic_Patch()
```

oneDay_oviposit_stochastic_Patch *Stochastic Oviposition*

Description

Calculate the number of eggs oviposited by female mosquitoes following:

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

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

Usage

```
oneDay_oviposit_stochastic_Patch()
```

oneDay_PopDynamics_Patch

Daily Population Dynamics for a Patch

Description

Run population dynamics (including migration) for this patch.
Performed in this order, see the following for each function:
Adult Death: `oneDay_adultDeath_deterministic_Patch` or `oneDay_adultDeath_stochastic_Patch`
Pupa Death/Maturation: `oneDay_pupaDM_deterministic_Patch` or `oneDay_pupaDM_stochastic_Patch`
Larva Death/Maturation: `oneDay_larvaDM_deterministic_Patch` or `oneDay_larvaDM_stochastic_Patch`
Egg Death/Maturation: `oneDay_eggDM_deterministic_Patch` or `oneDay_eggDM_stochastic_Patch`
Pupation: `oneDay_pupation_deterministic_Patch` or `oneDay_pupation_stochastic_Patch`
Releases: `oneDay_releases_Patch`
Mating: `oneDay_mating_deterministic_Patch` or `oneDay_mating_stochastic_Patch`
Lay Eggs: `oneDay_oviposit_deterministic_Patch` or `oneDay_oviposit_stochastic_Patch`
Release Eggs: `oneDay_eggReleases_Patch`

Usage

```
oneDay_PopDynamics_Patch()
```

oneDay_pupaDM_deterministic_Patch

Deterministic Pupa Death and Pupation

Description

Daily pupa survival is calculated according to

$$\overline{P_{[t-1]}} * (1 - \mu_{aq})$$

, where μ_{aq} corresponds to daily non-density-dependent aquatic mortality.
See `parameterizeMGDrivE` for how these parameters are derived.

Usage

```
oneDay_pupaDM_deterministic_Patch()
```

`oneDay_pupaDM_stochastic_Patch` *Stochastic Pupa Death and Pupation*

Description

Daily pupa survival is sampled from a binomial distribution, where survival probability is given by

$$1 - \mu_{aq}$$

. μ_{aq} corresponds to daily non-density-dependent aquatic mortality.
See `parameterizeMGDrivE` for how these parameters are derived.

Usage

```
oneDay_pupaDM_stochastic_Patch()
```

`oneDay_pupation_deterministic_Patch` *Deterministic Pupation*

Description

Pupa first undergo one extra day of survival, calculated as

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

. This is an artifact of the conversion from continuous to discrete time (as mentioned in the original Hancock paper this model is derived from).
Then, pupation into adult males is calculated as

$$(1 - \overline{\phi}) * \overline{P_{[t]}}$$

and into adult females as

$$\overline{\phi} * \overline{P_{[t]}}$$

Usage

```
oneDay_pupation_deterministic_Patch()
```

oneDay_pupation_stochastic_Patch *Stochastic Pupation*

Description

Pupa first undergo one extra day of survival, calculated as a binomial over

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

. This is an artifact of the conversion from continuous to discrete time (as mentioned in the original Hancock paper this model is derived from).
Then, pupation is sampled from a binomial, where $(1 - \overline{\phi})$ is the genotype-specific probability of becoming male, and $\overline{\phi}$ is the genotype-specific of becoming female.

Usage

```
oneDay_pupation_stochastic_Patch()
```

oneDay_releases_Patch *Release Male/Female/Mated-Female Mosquitoes in a Patch*

Description

Based on this patch's release schedule, generateReleaseVector, this function handles daily releases.

Usage

```
oneDay_releases_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(verbose = TRUE)
```

Arguments

`verbose`	Chatty? Default is TRUE

`parameterizeMGDrivE` *parameterizeMGDrivE*

Description

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

Usage

```
parameterizeMGDrivE(
  runID = 1L,
  nPatch,
  simTime,
  sampTime = 1L,
  tEgg = 1L,
  tLarva = 14L,
  tPupa = 1L,
  beta = 32,
  muAd = 0.123,
  popGrowth = 1.096,
  AdPopEQ,
  LarPopRatio,
  AdPopRatio_F,
  AdPopRatio_M,
  inheritanceCube
)
```

Arguments

runID	Begin counting runs with this set of parameters from this value
nPatch	Number of Patch
simTime	Maximum time to run simulation
sampTime	Times to sample, used as tNow %% sampTime, default is every day
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	Single number or vector of adult population size at equilibrium (single number implies all patches have the same population)
LarPopRatio	May be empty; if not, a vector gives the wildtype gene frequencies among larval stages at the beginning of simulation or a matrix provides different initial frequencies for each patch (every row is a different patch, must have nrow = nPatch)
AdPopRatio_F	May be empty; if not, a vector gives the wildtype gene frequencies among adult females at the beginning of simulation or a matrix provides different initial frequencies for each patch (every row is a different patch, must have nrow = nPatch)
AdPopRatio_M	May be empty; if not, a vector gives the wildtype gene frequencies among adult males at the beginning of simulation or a matrix provides different initial frequencies for each patch (every row is a different patch, must have nrow = nPatch)
inheritanceCube	Inheritance cube to check/set population ratios at the beginning of the simulation

Examples

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

Patch *Patch Class Definition*

Description

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

Format

An R6Class generator object

Constructor

- patchID: integer ID of this patch
- genotypesID: character vector of genotypes
- timeAq: integer vector of length 3 specifying the length of each aquatic stage
- numPatches: integer, total number of patches in this simulation
- adultEQ: integer, total adult population in this patch for the duration of the simulation
- larvalEQ: integer, total larval population in this patch for the duration of the simulation
- muAq: double vector, length 3, daily death rate for each aquatic stage
- alpha: double, density-dependent centering parameter, see `parameterizeMGDrivE`
- adultRatioF: named double vector, distribution of adult female genotypes, see `parameterizeMGDrivE`
- adultRatioM: named double vector, distribution of adult male genotypes, see `parameterizeMGDrivE`
- larvalRatio: named double vector, distribution of all aquatic genotypes, see `parameterizeMGDrivE`
- eggReleases: egg release schedule for this patch, see `basicRepeatedReleases`
- maleReleases: male release schedule for this patch, see `basicRepeatedReleases`
- femaleReleases: female release schedule for this patch, see `basicRepeatedReleases`
- matedFemaleReleases: mated females release schedule for this patch, see `basicRepeatedReleases`

Methods

- set_NetworkPointer: see `set_NetworkPointer_Patch`
- get_femalePopulation: see `get_femalePop_Patch`
- get_malePopulation: see `get_malePop_Patch`
- initialPopulation: see `set_initialPopulation_Patch`
- setPopulation: see `set_population_deterministic_Patch` or `set_population_stochastic_Patch`
- reset: see `reset_Patch`
- oneDay_initOutput: see `oneDay_initOutput_Patch`
- oneDay_writeOutput: see `oneDay_writeOutput_Patch`
- oneDay_migrationIn: see `oneDay_migrationIn_Patch`
- oneDay_PopDynamics: see `oneDay_PopDynamics_Patch`
- oneDay_adultD: see `oneDay_adultDeath_deterministic_Patch` or `oneDay_adultDeath_stochastic_Patch`
- oneDay_pupaDM: see `oneDay_pupaDM_deterministic_Patch` or `oneDay_pupaDM_stochastic_Patch`
- oneDay_larvaDM: see `oneDay_larvaDM_deterministic_Patch` or `oneDay_larvaDM_stochastic_Patch`
- oneDay_eggDM: see `oneDay_eggDM_deterministic_Patch` or `oneDay_eggDM_stochastic_Patch`
- oneDay_pupation: see `oneDay_pupation_deterministic_Patch` or `oneDay_pupation_stochastic_Patch`
- oneDay_releases: see `oneDay_releases_Patch`
- oneDay_releaseEggs: see `oneDay_eggReleases_Patch`
- oneDay_mating: see `oneDay_mating_deterministic_Patch` or `oneDay_mating_stochastic_Patch`
- oneDay_layEggs: see `oneDay_oviposit_deterministic_Patch` or `oneDay_oviposit_stochastic_Patch`

Fields

- patchID: integer ID of this patch
- popAquatic: matrix, nGenotype x sum(timeAquatic), holding all eggs, larva, and pupa
- popMale: vector, nGenotype x 1, holds adult males
- popFemale: matrix, nGenotype x nGenotype, holds mated adult females
- popHolder: vector, nGenotype x 1, temporary population storage
- popPupSex: vector, nGenotype x 1, used in stochastic pupation as another temporary population
- popUnmated: vector, nGenotype x 1, holds unmated females
- popAquatict0: matrix, nGenotype x sum(timeAquatic), holding all eggs, larva, and pupa for reset, see `reset_Patch`
- popMalet0: vector, nGenotype x 1, holds adult males for reset see `reset_Patch`
- popFemalet0: matrix, nGenotype x nGenotype, holds mated adult females for reset see `reset_Patch`
- eggReleases: list of egg releases for this patch. See `oneDay_eggReleases_Patch`
- maleReleases: list of adult male releases for this patch. See `oneDay_releases_Patch`
- femaleReleases: list of adult female releases for this patch. See `oneDay_releases_Patch`
- matedFemaleReleases: list of mated adult female releases for this patch. See `oneDay_releases_Patch`
- NetworkPointer: a reference to enclosing `Network`

`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)
```

`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.

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_initialPopulation_Patch` *Set Initial Population*

Description

This hidden function distributes the population at time 0 in the steady-state conformation. This involves finding the number of mosquitoes in each day of the aquatic stages, and then splitting adults into male and female. Each stage is appropriately split amongst the initial population genotypes (see `parameterizeMGDrivE`). It internally calls `calcLarvalDist` to determine the distribution of larvae before setting the eggs and pupa from that.

Usage

```
set_initialPopulation_Patch(
  adultEQ = adultEQ,
  larvalEQ = larvalEQ,
  adultRatioF = adultRatioF,
  adultRatioM = adultRatioM,
  larvalRatio = larvalRatio,
  timeAq = timeAq,
  muAq = muAq,
  alpha = alpha
)
```

Arguments

`adultEQ`	Equilibrium number of adults
`larvalEQ`	Equilibrium number of larvae
`adultRatioF`	Genotype specific ratio for adult females
`adultRatioM`	Genotype specific ratio for adult males
`larvalRatio`	Genotype specific ratio for larvae
`timeAq`	Time for each aquatic stage
`muAq`	Aquatic death rate
`alpha`	Density-dependent centering parameter

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

set_population_deterministic_Patch

Set Initial Population Deterministic

Description

Calls set_initialPopulation_Patch to initialize a steady-state population distribution.

Usage

```
set_population_deterministic_Patch(
  adultEQ = adultEQ,
  larvalEQ = larvalEQ,
  adultRatioF = adultRatioF,
  adultRatioM = adultRatioM,
  larvalRatio = larvalRatio,
  timeAq = timeAq,
  muAq = muAq,
  alpha = alpha
)
```

Arguments

adultEQ	Equilibrium number of adults
larvalEQ	Equilibrium number of larvae
adultRatioF	Genotype specific ratio for adult females
adultRatioM	Genotype specific ratio for adult males
larvalRatio	Genotype specific ratio for larvae

`timeAq`	Time for each aquatic stage
`muAq`	Aquatic death rate
`alpha`	Density-dependent centering parameter

---

set_population_stochastic_Patch

Set Initial Population Stochastic

Description

Calls `set_initialPopulation_Patch` to initialize a steady-state population distribution. Populations are then rounded to integer values.

Usage

```
set_population_stochastic_Patch(
  adultEQ = adultEQ,
  larvalEQ = larvalEQ,
  adultRatioF = adultRatioF,
  adultRatioM = adultRatioM,
  larvalRatio = larvalRatio,
  timeAq = timeAq,
  muAq = muAq,
  alpha = alpha
)
```

Arguments

`adultEQ`	Equilibrium number of adults
`larvalEQ`	Equilibrium number of larvae
`adultRatioF`	Genotype specific ratio for adult females
`adultRatioM`	Genotype specific ratio for adult males
`larvalRatio`	Genotype specific ratio for larvae
`timeAq`	Time for each aquatic stage
`muAq`	Aquatic death rate
`alpha`	Density-dependent centering parameter

splitOutput *Split Output by Patch*

Description

Split output into multiple files by patches.

Usage

```
splitOutput(readDir, writeDir = NULL, remFile = TRUE, 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
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
#  no return value
splitOutput(readDir = fPath, writeDir = oPath, remFile = FALSE)

# 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)

## End(Not run)
```

Index