Package 'IPMpack'

February 19, 2015

Type Package

Title Builds and analyses Integral Projection Models (IPMs).

Version 2.1

Date 2012-7-4

Author CJE Metcalf, SM McMahon, R Salguero-Gomez, E Jongejans, C Merow

Maintainer Sean McMahon <ipmpack@gmail.com>

Description IPMpack takes demographic vital rates and (optionally) environmental data to build integral projection models. A number of functional forms for growth and survival can be incorporated, as well as a range of reproductive strategies. The package also includes a suite of diagnostic routines, provides classic matrix model output (e.g., lambda, elasticities, sensitivities), and produces post-hoc metrics (e.g., passage time and life expectancy).

License GPL

LazyLoad yes

Depends Matrix, MASS, nlme

Suggests MCMCglmm, truncnorm, mvtnorm, methods, MCMCpack, fields

NeedsCompilation no

Repository CRAN

Date/Publication 2014-03-17 03:08:17

R topics documented:

`IPMpack-package` *Construction and analysis of integral projection models and associated measures of population growth, structure, perturbations (sensitivities and elasticities), overall population dynamics, age-specific metrics, etc.*

Description

IPM package, a series of analytical tools for building, diagnosing, and projecting populations models based on continuous and descrete vital rates.

Details

Package:	IPMpack
Type:	Package
Version:	1.3
Date:	2012-June-12
License:	GPL
LazyLoad:	yes

Depends:	MCMCglmm, Matrix, MASS, nlme

Author(s)

The IPMpack team: C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow. Maintainer: ipmpack-users@lists.r-forge.r-project.org; ipmpack@gmail.com User mailing list: https://lists.r-forge.r-project.org/mailman/listinfo/ipmpack-users

---

`addPdfGrowthPic` *Adds probability density functions of density function of size or increment given current size and growth to plots.*

Description

Function generates pdfs (probability density functions) corresponding to chosen sizes and adds them to a figure using growth methods.

Usage

```
addPdfGrowthPic(respType = "sizeNext", sizesPlotAt = c(20, 50, 60),
                               sizeRange = c(20, 400), incrRange = c(-10, 50),
                               scalar = 100, growthObjList,
                               cols = 1:5,
                               cov = data.frame(covariate=1),
                               minShow = 1e-2,
                               jitt = 2,
                               ...)
```

Arguments

`respType`	character string identifying the response variable for the growthModelComp. wither `"sizeNext"`, `"logincr"` or `"incr"`. Defaults to `sizeNext`.
`sizesPlotAt`	vector, list of sizes at which pdfs should be plotted.
`sizeRange`	sizeRange for which pdf should be estimated
`incrRange`	increment range for which pdf should be estimated
`scalar`	value by which pdf may be multiplied to improve visibility
`growthObjList`	list of growth objects for which pdfs are desired to be plotted
`cols`	colours corresponding to the list of growth objects for plotting
`cov`	a data-frame with one row containing all covariates other than size related covariates; defaults to 1; will be ignored if no covariates are fitted

`minShow`	minimum value below which pdf lines will not be shown (to avoid ugly vertical lines)
`jitt`	amount by which sequential pdfs should be separated on the x axis, for visibility
`...`	extra arguments to pass to the plot function.

Value

none

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

See Also

`makeGrowthObj`,`makeSurvObj`,`plotGrowthModelComp`, `plotSurvModelComp`

Examples

```
# Data with size and sizeNext
dff <- generateData()

a1 <- growthModelComp(dff, makePlot = TRUE)
addPdfGrowthPic(respType = "sizeNext",
sizesPlotAt = c(2, 6, 10), scalar = 1, jitt = 0.1,
sizeRange = c(-5, 25),
growthObjList = a1$growthObjects, cols = 2:5)
```

`coerceGrowthObj` *Function to coerce growth or survival objects, i.e., impose user-defined parameters*

Description

Supplied with a growth and survival object, over-writes coefficients, and for growth, the sd of growth

Usage

```
coerceGrowthObj(growthObj, coeff, sd)
coerceSurvObj(survObj,coeff)
```

Arguments

`growthObj`	an object of class growthObj
`survObj`	an object of class survObj
`coeff`	a numeric vector
`sd`	a numeric vector of length 1

Details

These functions can be used to impose coefficients and sd on growth and survival objects where direct fitting is not desired

Value

an object of class growthObj / survObj

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

See Also

`makeSurvObj`, `makeGrowthObj`

Examples

```
dff<-generateData()

#for growth
gr1 <- makeGrowthObj(dataf=dff,
Formula=sizeNext~size,regType="constantVar")

#halve the slope
gr2 <- coerceGrowthObj(gr1,coeff=c(gr1@fit$coefficients[1],
    gr1@fit$coefficients[2]*0.5),
    sd=gr1@sd)

par(mfrow=c(1,2),pty="s")
picGrow(dff,gr1)
picGrow(dff,gr2)

#for survival
sv1 <- makeSurvObj(dataf=dff,
Formula=surv~size)

#halve the slope
sv2 <- coerceSurvObj(sv1,coeff=c(sv1@fit$coefficients[1],
    sv1@fit$coefficients[2]*0.5))

par(mfrow=c(1,2),pty="s")
```

```
picSurv(dff,sv1)
picSurv(dff,sv2)
```

`convergeIPM`	*Iterates until obtaining the number of bins required so that the difference in the chosen measure (lambda, R0, life expectancy of a chosenBin) falls below a chosen tolerance level*

Description

Increases bin number by a specified `binIncrease` until the difference in the chosen measure falls below a tolerance level

Usage

```
convergeIPM(growObj, survObj, fecObj, nBigMatrix, minSize, maxSize,
discreteTrans = 1, integrateType = "midpoint",
correction = "none",  preCensus = TRUE, tol=1e-4,
binIncrease=5, chosenBin=1, response="lambda")
```

Arguments

`growObj`	a growth object.
`survObj`	a survival object.
`fecObj`	a fecundity object.
`nBigMatrix`	numeric, initial number of bins of size used in the matrix - will be increased for the assessment
`minSize`	numeric, minimum size used for meshpoints
`maxSize`	numeric, maximum size used for meshpoints of the P matrix.
`discreteTrans`	matrix of discrete transitions; or 1 if there is none
`integrateType`	integration type.
`correction`	correction (see makeIPMPmatrix)
`preCensus`	boolean defining whether fecundity is pre or post census; defaults to pre
`tol`	desired tolerance level
`binIncrease`	increments in increase in the number of bins (should be an integer)
`chosenBin`	desired bin for which life expectancy should be assessed; default is 1st.
`response`	what variable is convergence to be tested for; options are "lambda", "R0", "lifeExpect"; for the latter, the desired bin should be considered

Details

Different choices for responses will yield different values. The pattern of change in lambda (or other response variables) can be complex, so it is advisable to start with large binIncrease and small tolerance, and then once one knows a general idea of how big the matrix needs to be, run the function again with a smaller binIncrease but start it closer to the goal.

For the life expectancy option, if discrete stages are included via discreteTrans then if chosenBin=1, this function will use the first discrete bin.

Value

`binIncrease`	the number of bins used to increase matrix size in assessing tolerance
`Pmatrix`	the final Pmatrix if only LE is being considered
`IPM`	the final IPM
`R0`	the final R0
`lambda`	the final lambda
`LE`	the final vector of life expectancies

Note

This code was modified from original code by Melissa Eitzel.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

See Also

`diagnosticsPmatrix`

Examples

```
dff<-generateData()
gr1<-makeGrowthObj(dff)
sv1<-makeSurvObj(dff)
fv1<-makeFecObj(dff,Transform="log")

res <- convergeIPM(growObj=gr1,
 survObj=sv1, fecObj=fv1,
 nBigMatrix=10, minSize=-2,
 maxSize=15,discreteTrans = 1,
 integrateType = "midpoint",
 correction = "none",
 preCensus = TRUE, tol=1e-3,binIncrease=10)

res <- convergeIPM(growObj=gr1,
survObj=sv1, fecObj=fv1,
 nBigMatrix=10, minSize=-2,
 maxSize=15,discreteTrans = 1,
 integrateType = "midpoint",
 correction = "none",
 preCensus = TRUE, tol=1e-3,
 binIncrease=10, response="R0")

res <- convergeIPM(growObj=gr1, survObj=sv1, fecObj=fv1,
 nBigMatrix=10, minSize=-2,
 maxSize=15,discreteTrans = 1,
 integrateType = "midpoint",
 correction = "none",
 preCensus = TRUE, tol=1e-3,binIncrease=10,
 response="lifeExpect")
```

`convertIncrement` *Convert size increment according to time elapsed between censuses.*

Description

Adjusts the intervals of census data that is not annual to report output on population dynamics on an annual basis.

Usage

```
convertIncrement(dataf, nYrs = 1)
```

Arguments

`dataf`	a dataframe with columns 'size' 'sizeNext', 'exactDate', 'exactDateNext'.
`nYrs`	the number of years between sequential measurements (i.e. if census interval is 5, nYrs = 5, if census interval is 3 times a year, nYrs = 0.333).

Details

In some data sets the time interval between census measurements can be different than one year. In some species demographic information is recorded several times within a year, while in others, particularly in "slow-living" species (e.g. trees) the census frequency is greater than one year and/or vary across intervals. This function takes a data frame `dataf` and uses columns with the term `extractData` to adjust the size increment to the number of years given in `nyears`. It defaults to annual.

Value

Returns the adjusted increments.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Sampling three times/year (nYrs = 0.333): Smith, Caswell & Mettler-Cherry. Stochastic flood and precipitation regimes and the population dynamics of a threatened floodplain plant. Ecological Applications 15, p1036-1052.

Sampling every five years (nYrs = 5): van Mantgem & Stepheson. 2005. The accuracy of matrix population model projections for coniferous trees in the Sierra Nevada, California. Journal of Ecology 93, p737-747.

`dataIPMpackCryptantha` *Cryptantha Perennial Dataset with Covariates*

Description

Demographic data of Cryptantha flava in the "Redfleet State Park", UT, USA. Life cycle, experimental design and data are described in Salguero-Gomez et al (2012). Data contains a subset of individuals from 2004 to 2010. Full dataset can be obtained upon request to the authors (salguero@demogr.mpg.de and bcasper@sas.upenn.edu).

Usage

```
data(dataIPMpackCryptantha)
```

Format

The format is: chr "dataIPMpackCryptantha"

Details

Data-frame with headings:

- ID: unique plant id (this file contains only a subset of all individuals)

- treatment: the full experimental design contain to droughts (in 1998 and 1999) but this subset contains only info on the control permanent plots. See Lucas et al J Ecol 2008

- site: spatial replication site

- plot: plot number

- quadrat: quadrat number inside of plot

- x: x coordinate (cm) inside of the quadrat

- y: y coordinate (cm) inside of the quadrat

- shrub: shrub species within the zone of influence of the individual (At = Artemisia tridentata; Cn = Chrysothamnus nauseosus)

- compass: compass direction of the line connecting the centroid of the shrub and the individual of Cryptantha flava

- distance: distance (in cm) of the individual of Cryptantha flava to the shrub. Negative distance imply the individual is "inside" the shrub. Zero implies the individual is at the edge of the shrub's canopy.

- year: transition from t to t+1 (this subset contains only data for 2004-2010)

- prec: annual precipitation (in cm) from June of year t-1 to May of year t

- age: age of individual (in years) in year t. Individuals of unknown age are assigned to 999

- size: total number of rosettes (vegetative and flowering) of the individual in year t

- fec0: probability of reproduction (0: vegetative, 1: flowering; NA: individual not alive) in year t

- fec1: number of flowering rosettes in year t (NA: fec0 = 0 or NA)

- surv: survival (0 = dead, 1= alive, NAs if not yet recruited)

- precNext: annual precipitation (in cm) from June of year t to May of year t+1

- ageNext: age of individual (in years) in year t+1. Individuals of unknown age are assigned to 999

- sizeNext: total number of rosettes (vegetative and flowering) of the individual in year t+1

Author(s)

Rob Salguero-Gomez & Brenda B Casper

References

Salguero-Gomez R, Siewert W, Casper B & Tielboerger K. Oct 2012. A demographic approach to study effects of climate change in desert plants. Philosophical Transactions of the Royal Society. Series B - Biological Sciences x, pxxx-xxx

Lucas R, Forseth I, Casper B. 2008. Using rainout shelters to evaluate climate change effects on the demography of Cryptantha flava. Journal of Ecology 96, p514-522

Examples

```
data(dataIPMpackCryptantha)
print(head(dataIPMpackCryptantha))
```

`dataIPMpackHypericum` *Hypericum Perennial Dataset*

Description

Demographic data of Hypericum cumulicola in "Florida rosemary scrub at" Archbold Biological Station (FL, USA). Life cycle, experimental design and data are described in Quintana-Ascencio & Menges (2003). Data contains a subset of individuals from population "bald 1" and annual period "1997-1998". Full dataset can be obtained upon request to the authors (pedro.quintana-ascencio@ucf.edu).

Usage

```
data(dataIPMpackHypericum)
```

Format

The format is: chr "dataIPMpackHypericum"

Details

data-frame with headings:

- id: unique plant id (this file contains only a subset of all individuals)

- bald: population (this subset contains only one population)

- year: transition from t to t+1 (this subset contains only data for 1997-1998)

- size: length of longest stem in individual (cm) in time t

- ontogeny: recruits vs established individuals in time t (1 = individual was recruited in time t, 0 = already established individual prior to time t, NA = individual not yet recruited in time t)

- fec0: probability of reproduction (0= no flowering, 1 = individual was flowering in time t, NA = individual not alive in year t)

- fec1: number of fruits per plant (NA if fec0 = 0)

- surv: survival (0 = dead, 1= alive, NAs if not yet recruited or past dead)

- sizeNext: length of longest stem in individual (cm) in time t+1

- ontogenyNext: recruits vs established individuals in time t+1 (1 = individual was recruited in time t+1, 0 = already established individual prior to time t+1, NA = individual not yet recruited or dead in t+1)

Author(s)

Pedro Quintana Ascencio & Eric Menges

References

Quintana-Ascencio, Menges & Weekley. 2003. A fire-explicit population viability analyses of Hypericum cumulicola in Florida Rosemary scrub. Conservation Biology 17, p433-449.

Examples

```
#Access data from the long-term censuses on Hypericum cumulicula
# carried out by Eric Menges, Pedro Quintana-Ascencio and coworkers
# at Archbold Biological Station. Here only a subset of individuals
# from population 'bald 1' and for the annual transition '1997-1998' are shown.
data("dataIPMpackHypericum")
d<-dataIPMpackHypericum

#Variables are:
  #id: unique identifier for each individual
  #bald: population. Here only bald 1
  #year: annual transition of the long-term data. Here only 1997-1998
  #surv: survival (1) or not (0) of individuals between 1997 and 1998
  #size: maximum height of the stems of each individual
  #ontogeny: because the demography of Hypericum is very dynamic
  #      (turnover is very high) the experimental design described
  #       in Quintana-Ascencio et al. (2003) consists on establishing
  #       new permanent plots every year at each population,
  #       in addition to censusing old plots. Here we differentiate
  #       between individuals that appear for the first time in time t
  #       because they were recruits (1) and those that, not being new
  #       recruits, where measured for the first time in t because they
  #       were in a new permanent plot.
  #fec0: probability of flowering (1) or not (0)
  #fec1: number of fruits per individual
  #sizeNext: same as "size" above, for t+1
  #stageNext: same as "stage" above, for t+1

#Due to the sampling design described above, here we consider only
# individual with a certain recruit origin:
d <- subset(d,is.na(d$size)==FALSE | d$ontogenyNext==1)

#Side-experiments revealed that the following vital rates are size-independent
# and equal to:
  #Number of seeds produced per fruit
    fec2<-13.78
  #Probability of seedling establishment
    fec3<-0.001336
  #Probability of seedling survival half a year after germinating,
  #   corresponding to the next annual census
    fec4<-0.14
  #Probability of a seed going into the seed bank
    goSB<-0.08234528
  #Probability of a seed staying in the seed bank
    staySB<-0.672
#Note that the aforementioned vital rates are function of time since last fire,
#but because here we are only dealing with one population and one year
#transition, we treat them as constants. See Quintana-Ascencio et al (2003)
#for more information.

#A simple re-organization of the data, getting rid of non-critical information
d<-d[,c("surv","size","sizeNext","fec0","fec1")]

#The following states the continuous (max height of individual plant)
#part of the IPM. Note that the IPM to be constructed here contains a
#discrete stage: seedbank.
d$stageNext<-d$stage<-"continuous"
d$stage[is.na(d$size)]<-NA
#If individual did not survive, it is labelled as dead to t+1.
d$stageNext[d$surv==0]<-"dead"
#Adds probability of seeds going into (continuous -> seedbank),
#staying (seedbank -> seedbank) and leaving (continuous -> seedbank)
#the discrete stage.
d$number<-1
```

```
d$stage<-as.factor(d$stage)
d$stageNext<-as.factor(d$stageNext)

#Carry out comparisons to establish the best survival model
testSurv <- survModelComp(d, expVars = c(surv~1, surv~size,
 surv~size + size2), testType = "AIC",makePlot = TRUE,legendPos = "bottomleft")

#Carry out comparisons to establish the best growth model
testGrow <- growthModelComp(d,expVars = c(sizeNext~1, sizeNext~size,
   sizeNext~size + size2), regressionType = "constantVar",
   testType = "AIC", makePlot = TRUE, legendPos = "bottomright")

#Create survival object using regression model indicated by testSurv
so <- makeSurvObj(d, Formula = surv~size + size2)
picSurv(d,so)

#Create growth object using regression model indicated by testGrown
go<-makeGrowthObj(d, Formula = sizeNext~size)
picGrow(d,go)
abline(a=0,b=1,lty=2)

#Create fecundity object using regression models
fo <- makeFecObj(d, Formula=c(fec0~size, fec1~size),
                 Family=c("binomial","poisson"),
                 Transform=c("none", "none"),
                 meanOffspringSize=mean(d[is.na(d$size)==TRUE &
                 is.na(d$sizeNext)==FALSE,"sizeNext"]),
                 sdOffspringSize=sd(d[is.na(d$size)==TRUE &
                 is.na(d$sizeNext)==FALSE,"sizeNext"]),
                 fecConstants=data.frame(fec2=fec2,fec3=fec3,fec4=fec4),
                 offspringSplitter=data.frame(seedbank=goSB,
                    continuous=(1-goSB)),
                 vitalRatesPerOffspringType=data.frame(seedbank=c(1,1,1,0,0),
                                              continuous=c(1,1,1,1,1),
                                              row.names=c("fec0","fec1",
                                              "fec2","fec3","fec4")))


#Define discrete transition matrix
dto<-makeDiscreteTrans(d,
discreteTrans = matrix(c(staySB,(1-staySB)*fec3*fec4,
(1-staySB)*(1-fec3*fec4),0,
sum(d$number[d$stage=="continuous"&d$stageNext=="continuous"],
na.rm=TRUE),sum(d$number[d$stage=="continuous"&d$stageNext=="dead"],
na.rm=TRUE)),ncol=2,nrow=3,
dimnames=list(c("seedbank","continuous","dead"),
c("seedbank","continuous"))),
meanToCont = matrix(mean(d$sizeNext[is.na(d$stage)&
d$stageNext=="continuous"]),ncol=1,nrow=1,dimnames=list(c("mean"),
c("seedbank"))),
sdToCont = matrix(sd(d$sizeNext[is.na(d$stage)&
d$stageNext=="continuous"]),ncol=1,nrow=1,dimnames=list(c(""),
c("seedbank"))))
```

```
#choose number of bins for discretization in the IPM
nBigMatrix <- 100

#Create the P matrix describing growth-survival transitions
#  The argument correction="discretizeExtremes" places parts of the
# growth distribution that fall
#  below minSize or above maxSize into the first and last bin
#
Pmatrix<-makeIPMPmatrix(growObj=go,survObj=so,discreteTrans=dto,
                          minSize=0,maxSize=80,nBigMatrix=nBigMatrix,
                          correction="discretizeExtremes")

#Create the F matrix descributing fecundity transitions
#  The argument correction="discretizeExtremes" places parts of the
#  continuous offspring distribution that fall
#  below minSize or above maxSize into the first and last bin
#
Fmatrix<-makeIPMFmatrix(fecObj=fo,
                          minSize=0,maxSize=80,nBigMatrix=nBigMatrix,
                          correction="discretizeExtremes")

#Build a P matrix reflecting only the continuous part of the model
# and check that binning, etc is adequate
PmatrixContinuousOnly <- makeIPMPmatrix(growObj=go,
    survObj=so,minSize=0,maxSize=70,nBigMatrix=nBigMatrix,
        correction="discretizeExtremes")
diagnosticsPmatrix(PmatrixContinuousOnly,growObj=go,
    survObj=so,dff=d, correction="discretizeExtremes")

#Form the IPM as a result of adding the P and F matrices
IPM <- Pmatrix + Fmatrix

#Population growth rate for the whole life cycle of Hypericum is
eigen(IPM)$value[1]
#Population growth rate excluding the seed bank stage is
eigen(IPM[2:(nBigMatrix+1),2:(nBigMatrix+1)])$value[1]
```

dataIPMpackHypericumCov

Hypericum Perennial Dataset with covariates

Description

Demographic data of Hypericum cumulicola in "Florida rosemary scrub at" Archbold Biological Station (FL, USA). Life cycle, experimental design and data are described in Quintana-Ascencio & Menges (2003). Data contains a subset of individuals from population "bald 1" and annual period "1994-1999". Full dataset can be obtained upon request to the authors (pedro.quintana-ascencio@ucf.edu).

Usage

```
data(dataIPMpackHypericumCov)
```

Format

The format is: chr "dataIPMpackHypericumCov"

Details

data-frame with headings:

- id: unique plant id (this file contains only a subset of all individuals)

- bald: population (this subset contains only one population)

- year: transition from t to t+1 (this subset contains only data for 1994-1999)

- fireYear: year when the bald was last burned

- TSLF: time since last fire

- size: length of longest stem in individual (cm) in time t

- ontogeny: recruits vs established individuals in time t (1 = individual was recruited in time t, 0 = already established individual prior to time t, NA = individual not yet recruited in time t)

- fec0: probability of reproduction (0= no flowering, 1 = individual was flowering in time t)

- fec1: number of fruits per plant (NA if fec0 = 0)

- fec2: number of seeds per fruit (NA if fec0 = 0)

- fec3: probability that seeds produced in year t will germinate that year

- fec4: probability of seedling survival prior to the next census

- goSB: probability that seeds produced in natural year t will go into the seedbank

- staySB: probability that seeds will remain in the seedbank from year t to t+1

- cov: annual precipitation in year t (Jan-Dec; mm)

- surv: survival (0 = dead, 1= alive, NAs if not yet recruited)

- sizeNext: length of longest stem in individual (cm) in time t+1

- ontogenyNext: recruits vs established individuals in time t+1 (1 = individual was recruited in time t+1, 0 = already established individual prior to time t+1, NA = individual not yet recruited or dead in t+1)

- covNext: annual precipitation in natural year t+1 (Jan-Dec; mm)

Author(s)

Pedro Quintana-Ascencio & Eric Menges

References

Quintana-Ascencio, Menges & Weekley. 2003. A fire-explicit population viability analyses of Hypericum cumulicola in Florida Rosemary scrub. Conservation Biology 17, p433-449.

Examples

```
data(dataIPMpackHypericumCov)
print(head(dataIPMpackHypericumCov))
```

`dataIPMpackSilwood` *Silwood Monocarp Dataset*

Description

Demographic data of several monocarpic plants from Silwood (UK)

Usage

```
data(dataIPMpackSilwood)
```

Format

The format is: chr "dataIPMpackSilwood"

Details

data-frame with headings:

- exactDate: date measurement

- exactDateNext: date next measurement

- id: unique plant id

- Species: species name

- Site: location within Silwood park where plant was measured

- rtcr: root crown diameter, measured with caliper (cm)

- rtcrNext: root crown diameter at next census time (cm)

- ll: length of longest leaf (cm)

- llNext: length of longest leaf at next census (cm)

- rosetteDiam: rosette Diameter (cm()

- rosetteDiamNext: rosette diameter at next census (cm)

- flowered: probability of reproduction (0: individual did not flowered, 1: individual flowered)

- surv: survival (0: dead, 1: survival, NA: not known)

References

Data used in: Metcalf, C.J.E., Rees, M., Alexander, J.M., Rose, K.E. 2006. Growth-survival trade-offs and allometries in rosette-forming perennials. Funct. Ecol. 20, 217-225.

Examples

```
data(dataIPMpackSilwood)
print(head(dataIPMpackSilwood))
plot(dataIPMpackSilwood$rtcr,dataIPMpackSilwood$rtcrNext,
xlab="size now", ylab="size next", pch=19,log="xy")
## maybe str(dataIPMpackSilwood) ; plot(dataIPMpackSilwood) ...
```

`dataIPMpackSuccisa` *Succisa pratensis Dataset*

Description

Subset of multiple years of demographic data of Succisa pratensis collected at the "Bennekomse Meent" site in the Netherlands. Details are described in Jongejans and de Kroon (2005). More information: E.Jongejans@science.ru.nl

Usage

```
data(dataIPMpackSuccisa)
```

Format

The format is: chr "dataIPMpackSuccisa"

Details

data-frame with headings:

- size: log of the product of the number of leaves and the maximum leaf length of rosette leaves at time t

- sizeNext: log of the product of the number of leaves and the maximum leaf length of rosette leaves at time t+1

- stage: stage of the individual ('continuous' means the plant has a size; NA = not recruited yet) at time t

- stageNext: stage of the individual ('continuous' means the plant has a size; 'dead' = dead) at time t+1

- surv: survival (0 = dead, 1= alive, NAs if not yet recruited)

- offspringNext: type of new recruit ('sexual' = seedling, 'clonal' = side-rosette, NA = not a recruit) at time t+1

- fec1Bolt: whether stems are produced (0 = no stems, 1 = at least 1 stem, NA = not recruited yet) at time t

- fec2Stem: number of stems when stems are produced (NA when fec1Bolt = NA or 0) at time t

- fec3Head: mean number of flower heads per stem when stems are produced (NA when fec1Bolt = NA or 0) at time t

- fec1BoltNext: whether stems are produced (0 = no stems, 1 = at least 1 stem, NA = not recruited yet) at time t+1

- fec2StemNext: number of stems when stems are produced (NA when fec1BoltNext = NA or 0) at time t+1

- fec3HeadNext: mean number of flower heads per stem when stems are produced (NA when fec1BoltNext = NA or 0) at time t+1

- cloning: whether clonal offspring (side-rosettes) are produced by this individual (NA for plants that were not recruited yet at time t)

- clonesNext: number of clonal offspring (side-rosettes) produced when at least on side-rosette is produced (NA when 'cloning' = 0 or NA)

Author(s)

Eelke Jongejans & Hans de Kroon

References

Jongejans, E. and de Kroon, H. (2005) Space versus time variation in the population dynamics of three co-occurring perennial herbs. Journal of Ecology, 93, 681-692.

Examples

```
data(dataIPMpackSuccisa)
print(head(dataIPMpackSuccisa))

Sp <- dataIPMpackSuccisa

fo<-makeFecObj(Sp, Formula = list(fec1Bolt ~ size+size2,
    fec2Stem ~ size, fec3Head ~ size),
       Family = c("binomial","poisson","poisson"),
       Transform=c("none","-1","none"),
       fecConstants = data.frame(seedsPerHead=50,
       seedlingEstablishmentRate= 0.02))

co<-makeClonalObj(Sp, Formula = list(cloning ~ size,
    clonesNext ~ size), Family = c("binomial","poisson"),
        Transform=c("none","-1"), offspringSizeExplanatoryVariables = "size")
```

`dataIPMpackSuccisa2` *Succisa pratensis Dataset*

Description

Subset of multiple years of demographic data of Succisa pratensis collected at the "Bennekomse Meent" site in the Netherlands. Details are described in Jongejans and de Kroon (2005). More information: E.Jongejans@science.ru.nl

Usage

```
data(dataIPMpackSuccisa2)
```

Format

The format is: chr "dataIPMpackSuccisa2"

Details

data-frame with headings:

- size: log of the sum of the products of the number of leaves and the maximum leaf length for rosette and stem leaves at time t

- sizeNext: log of the sum of the products of the number of leaves and the maximum leaf length for rosette and stem leaves at time t+1

- stage: stage of the individual ('continuous' means the plant has a size; NA = not recruited yet) at time t

- stageNext: stage of the individual ('continuous' means the plant has a size; 'dead' = dead) at time t+1

- surv: survival (0 = dead, 1= alive, NAs if not yet recruited)

- offspringNext: type of new recruit ('sexual' = seedling, 'clonal' = side-rosette, NA = not a recruit) at time t+1

- fec1Bolt: whether stems are produced (0 = no stems, 1 = at least 1 stem, NA = not recruited yet) at time t

- fec2Stem: number of stems when stems are produced (NA when fec1Bolt = NA or 0) at time t

- fec3Head: mean number of flower heads per stem when stems are produced (NA when fec1Bolt = NA or 0) at time t

- fec1BoltNext: whether stems are produced (0 = no stems, 1 = at least 1 stem, NA = not recruited yet) at time t+1

- fec2StemNext: number of stems when stems are produced (NA when fec1BoltNext = NA or 0) at time t+1

- fec3HeadNext: mean number of flower heads per stem when stems are produced (NA when fec1BoltNext = NA or 0) at time t+1

- cloning: whether clonal offspring (side-rosettes) are produced by this individual (NA for plants that were not recruited yet at time t)

- clonesNext: number of clonal offspring (side-rosettes) produced when at least on side-rosette is produced (NA when 'cloning' = 0 or NA)

Author(s)

Eelke Jongejans & Hans de Kroon

References

Jongejans, E. and de Kroon, H. (2005) Space versus time variation in the population dynamics of three co-occurring perennial herbs. Journal of Ecology, 93, 681-692.

Examples

```
data(dataIPMpackSuccisa2)
print(head(dataIPMpackSuccisa2))

Sp <- dataIPMpackSuccisa2

fo<-makeFecObj(Sp, Formula = list(fec1Bolt ~ size+size2,
    fec2Stem ~ size, fec3Head ~ size),
        Family = c("binomial","poisson","poisson"),
            Transform=c("none","-1","none"),
            fecConstants = data.frame(seedsPerHead=50,
            seedlingEstablishmentRate= 0.02))

co<-makeClonalObj(Sp, Formula = list(cloning ~ size, clonesNext ~ size),
    Family = c("binomial","poisson"), Transform=c("none","-1"),
       offspringSizeExplanatoryVariables = "size")
```

`diagnosticsPmatrix` *Creates a series of diagnostic graphs for a P matrix.*

Description

Displays the effects of increasing number of bins and continuous (size) stage range on a number of predictions from the P matrix to verify that sufficient resolution and continuous stage range are being used.

Usage

```
diagnosticsPmatrix(Pmatrix, growObj, survObj, dff=NULL, integrateType,
correction, cov = data.frame(covariate = 1), sizesToPlot = c(),
extendSizeRange = c())
```

Arguments

`Pmatrix`	an IPMmatrix object.
`growObj`	the growth object used to construct the IPMmatrix object.
`survObj`	the survival object used to construct the IPMmatrix object.
`dff`	the dataframe from which the survival and growth objects were constructed; if not supplied, defaults to NULL, which will simply result in the size distribution not being plotted.
`integrateType`	integration type, defaults to `midpoint` (which uses probability density function); other option is `cumul` (which uses the cumulative density function).
`correction`	correction type, defaults to `none`. The first option is `constant` which will multiply every column of the IPM by a constant sufficient to adjust values to those predicted for survival at that size. The second option is `discretizeExtremes` which will place all transitions to sizes smaller than `minSize` into the smallest bin, and transitions to sizes larger than `maxSize` into the largest bin.
`cov`	a data-frame with one row containing all covariates; defaults to 1, and will be ignored if covariates do not exist in growth and survival objects
`sizesToPlot`	a vector containing desired sizes to plot growth resolution for (second panel); if not supplied, the function will use the quantiles
`extendSizeRange`	a vector containing desired size range for the matrix to be compared with a larger size range; if this vector has length 0 the defaults will be 0.5xminSize (unless minSize<0 in which case, 2*minSize is used) and 1.5*maxSize

Details

This function provides a series of plots indicative of whether bin choice and size range is adequate. On the first plot, the left panel shows the range of the data as a histogram (if a data-frame is provided) and the range of the state variable fitted in the current Pmatrix; as well as the range of the state variable in two Pmatrices used for comparison, one with the same number of bins but an extended size range (red), and one with the same size range but an increased number of bins (blue) (increased by about 50 percent). If the range in the data and the range in the Pmatrix are mis-matched, the limits of the data used in building the Pmatrix can be adjusted with the `minSize` and `maxSize` arguments in `makeIPMPmatrix`.

The discretization of a continuous function can result in under- or over-estimation of the true density. Where this occurs, the sum of the columns of the discretized Pmatrix will not match predictions from the fitted survival model. The middle panel plots these against each other for the three matrices in the first panel (current, extended range and increased bin number) using the same colours as in the first panel. Lines should fall along the (0,1) line shown in grey; if they do not, the argument `correction="constant"` may be of use. This ensures that the columns sum to the fitted survival by multiplying every column in the Integral Projection Model by the value that allows this. The third panel checks whether extending the size range included in the matrix and increasing the number of bins (by increasing `nBigMatrix` and thereby having narrower bins) does not alter basic predictions from the IPM.

The six panels on the next plot show the discretized IPM (histograms) for the current IPM (top) and one with an increased number of bins (bottom) and the theoretical density function (red line)

. These are plotted either for three chosen sizes (`sizesToPlot`) or the 0.25, 0.5 and 0.75 quantiles of either the observed data or the range of meshpoints; this size is printed in the top right hand of every plot. If the theoretical density function curve is very distant from the histograms, increasing the `nBigMatrix` argument may correct this discrepancy.

Note that it is important that the comparison be "fair" - i.e., whatever the `correction` argument used in your current IPM, the same argument must be used in `diagnosticsPmatrix.`

Note also that if survival is constant across size, the patterns apparent in the life expectancy plot will reflect numerical slippage (since all sizes will have exactly the same life expectancy) and disagreement between the different lines should be ignored.

Finally, note that if the correction used is `discretizeExtremes` then the column sums of the Pmatrix and survival will not match towards the extremes.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Easterling, Ellner & Dixon. 2000. Size-specific sensitivity: a new structured population model. Ecology 81, p694-708.

Ellner & Rees. 2006. Integral projection models for species with complex demography. The American Naturalist 167, p410-428.

For effects of mesh size on IPM output: Zuidema, Jongejans, Chien, During & Schieving. Integral projection models for trees: a new parameterization method and a validation of model output. Journal of Ecology 98, p345-355.

See Also

`makeIPMPmatrix`, `convergeIPM`

Examples

```
# Example where mesh size does not have a major effect on model output:
dff <- generateData()
Pmatrix <- makeIPMPmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE), growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff))
diagnosticsPmatrix(Pmatrix, growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff), dff = dff)

# Compare with the following example where mesh size has an important
# effect on output:
Pmatrix <- makeIPMPmatrix(nBigMatrix = 8,
    minSize = min(dff$size, na.rm = TRUE),
maxSize = 0.5*max(dff$size, na.rm = TRUE),
   growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff))
diagnosticsPmatrix(Pmatrix, growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff), dff = dff)

#with cumul
Pmatrix <- makeIPMPmatrix(nBigMatrix = 10,
    minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE), growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff),
integrateType = "cumul")
diagnosticsPmatrix(Pmatrix, growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff), dff = dff, integrateType = "cumul")


#with log increment
Pmatrix <- makeIPMPmatrix(nBigMatrix = 50,
    minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE),
   growObj = makeGrowthObj(dff,Formula = logincr~size),
survObj = makeSurvObj(dff))
diagnosticsPmatrix(Pmatrix,
    growObj = makeGrowthObj(dff,Formula = logincr~size),
survObj = makeSurvObj(dff), dff = dff)

#example with correction="discretizeExtremes"
Pmatrix <- makeIPMPmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE), growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff), correction="discretizeExtremes")
diagnosticsPmatrix(Pmatrix, growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff), dff = dff, correction="discretizeExtremes")
```

discreteTrans-class   *Class* "discreteTrans"

Description

Matrix defining transitions between discrete stages; slots define the names of stages, etc.

Objects from the Class

Objects can be created by calls of the form `new("discreteTrans", ...)`.

Slots

`discreteTrans`: Object of class "matrix" ~~

`meanToCont`: Object of class "matrix" ~~

`sdToCont`: Object of class "matrix" ~~

`moveToDiscrete`: Object of class "glm" ~~

Methods

No methods defined with class "discreteTrans" in the signature.

Examples

```
showClass("discreteTrans")
```

`discreteTransInteger-class`
Class `"discreteTransInteger"`

Description

Matrix defining transitions between discrete stages; slots define the names of stages, etc.

Objects from the Class

Objects can be created by calls of the form `new("discreteTransInteger", ...)`.

Slots

`discreteTrans`**:** Object of class `"matrix"` ~~

`meanToCont`**:** Object of class `"matrix"` ~~

`thetaToCont`**:** Object of class `"matrix"` ~~

`moveToDiscrete`**:** Object of class `"glm"` ~~

`distToCont`**:** Object of class `"character"` ~~

Methods

No methods defined with class "discreteTrans" in the signature.

Examples

```
showClass("discreteTransInteger")
```

elas	*Estimates matrix element sensitivities and elasticities.*

Description

Estimates sensitivities and elasticities of each element of a discretized IPM.

Usage

```
elas(A)
sens(A)
```

Arguments

`A`	a matrix defining all transitions (growth, survival, fecundity) between sizes/stages.

Value

a matrix.

Note

Modified following code developed by Mark Rees & Dylan Childs

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Caswell, 2001, Matrix Population Models: construction, analysis, interpretation. 2nd ed. Sinauer. p206-256.

de Kroon, Plaisier, van Groenendael & Caswell. 1986. Elasticity: the relative contribution of demographic parameters to population growth rate. Ecology 67, p1427-1431.

de Kroon, van Groenendael & Ehrlen. 2000. Elasticities: a review of methods and model limitations. Ecology 81, p607-618.

See Also

`sens`, `sensParams`

Examples

```
dff <- generateData()
Pmatrix <- makeIPMPmatrix(minSize = min(dff$size, na.rm=TRUE),
maxSize = max(dff$size, na.rm = TRUE), growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff))

Fmatrix <- makeIPMFmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE), fecObj = makeFecObj(dff))

IPM <- Pmatrix + Fmatrix

par(mfrow = c(1, 2))

senst <- sens(IPM)
image(Pmatrix@meshpoints, Pmatrix@meshpoints,t(senst),
main = "Sensitivity", xlab = "Continuous (e.g. size) stage in t",
ylab = "Continuous (e.g. size) stage in t+1")

elast <- elas(IPM)
image(Pmatrix@meshpoints, Pmatrix@meshpoints, t(elast), main = "Elasticity",
xlab = "Continuous (e.g. size) stage in t",
ylab = "Continuous (e.g. size) stage in t+1")
```

`envMatrix-class` *Class "envMatrix"*

Description

A class object description

Objects from the Class

Objects can be created by calls of the form `new("envMatrix", data, nrow, ncol, byrow, dimnames, ...)`.

Slots

`.Data`: Object of class `"matrix"` ~~

`nEnvClass`: Object of class `"numeric"` ~~

Extends

Class `"matrix"`, from data part. Class `"array"`, by class "matrix", distance 2. Class `"structure"`, by class "matrix", distance 3. Class `"vector"`, by class "matrix", distance 4, with explicit coerce.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

Examples

```
showClass("envMatrix")
```

`fecObj-class` *Class "fecObj"*

Description

A class object description

Objects from the Class

Objects can be created by calls of the form `new("fecObj", ...)`.

Slots

"fecConstants" "offspringSplitter" "meanOffspringSize" "sdOffspringSize" "fecByDiscrete" "Transform"

Object of class `"list"` ~~

`fitFecConstants`**:** Object of class `"data.frame"` ~~

`fecNames`**:** Object of class `"character"` ~~

`offspringSplitter`**:** Object of class `"data.frame"` ~~

`offspringRel`**:** Object of class `"lm"` ~~

`vitalRatesPerOffspringType`**:** Object of class `"data.frame"` ~~

`sdOffspringSize`**:** Object of class `"numeric"` ~~

`fecByDiscrete`**:** Object of class `"data.frame"` ~~

`Transform`**:** Object of class `"character"` ~~

Methods

No methods defined with class "fecObj" in the signature.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

Examples

```
showClass("fecObj")
```

`fecObjInteger-class` *Class "fecObjInteger"*

Description

A class object description

Objects from the Class

Objects can be created by calls of the form `new("fecObjInteger", ...)`.

Slots

"fecConstants" "offspringSplitter" "meanOffspringSize" "sdOffspringSize" "fecByDiscrete" "Transform"

Object of class `"list"` ~~

`fitFecConstants`: Object of class `"data.frame"` ~~

`fecNames`: Object of class `"character"` ~~

`offspringSplitter`: Object of class `"data.frame"` ~~

`offspringRel`: Object of class `"glm"` ~~

`vitalRatesPerOffspringType`: Object of class `"data.frame"` ~~

`thetaOffspringSize`: Object of class `"numeric"` ~~

`fecByDiscrete`: Object of class `"data.frame"` ~~

`Transform`: Object of class `"character"` ~~

`distOffspring`: Object of class `"character"` ~~

Methods

No methods defined with class "fecObj" in the signature.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

Examples

```
showClass("fecObjInteger")
```

`generateData` *Generates random data in the form used by IPMpack.*

Description

Simulates growth, survival and fecundity to create a dataframe of the form required by the functions and methods used in IPMpack. Demographic stage data is only continuous.

Usage

```
generateData(nSamp=1000, type="simple")
```

Arguments

`nSamp`	number of samples desired in the base population, defaults to 1000
`type`	the kind of simulated data required. Supported values include "simple" (which includes only a continuous stage and a discretely varying covariate); "discrete" (which includes a discretely varying life stage); and "stochastic" (which includes stochastically varying covariates)

Value

A dataframe with headings: - "size": continuous variable, indicating current size. - "sizeNext" continuous variable, indicating size in the next time step. - "surv": boolean, indicating whether individual survived or not to the next time step. - "covariate": discrete covariate (for type="simple"). - "covariateNext": discrete covariate in the next (for type="simple") time step. - "covariate1", "covariate2", "covariate3", ...: discrete or continuous covariates (for type="stochastic".

- "fec": continuous variable, indicating fecundity. - "stage": character vector, containing names of the discrete stages in that time-step, or "continuous" (for type="discrete"). - "stageNext": character vector, containing names of the "discrete" stages in the following time-step, or "continuous" size value (for type="discrete"). - "number": number of individuals moving between stages. "number" = 1 for all movements out of the "continuous" stage; "number" > 1 for all movements out of "discrete" stages. This allows the user to not need to have an individual line for every movement between discrete stages (for type="discrete").

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

See Also

`simulateCarlina`

Examples

```
dff <- generateData(nSamp=2000, type="simple")
head(dff)

dff <- generateData(nSamp=2000, type="discrete")
head(dff)

dff <- generateData(nSamp=2000, type="stochastic")
head(dff)
```

`growSurv` *Combines growth and survival.*

Description

Predicts the probability density function of continuous (e.g. size) stage at time t+1 given stage values at time t and survival probability as a function of stage values at time t, given a growth and survival object.

Usage

```
growSurv(size, sizeNext, cov, growthObj, survObj)
```

Arguments

`size`	vector of current size(s).
`sizeNext`	vector of future size(s).
`cov`	covariate level (numeric of length 1).
`growthObj`	a growth object.
`survObj`	a survival object.

Details

`makeIPMPmatrix` and variants there-of apply `outer` to this function to efficiently obtain the IPM P matrix.

Value

numeric defining the pdf (probability density function).

Note

Code developed following Mark Rees, Dylan Childs & Karen Rose.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Easterling, Ellner & Dixon. 2000. Size-specific sensitivity: a new structured population model. Ecology 81, p694-708.

See Also

growth, surv

Examples

```
dff <- generateData()
gr1 <- makeGrowthObj(dff)
sv1 <- makeSurvObj(dff)
sizeRange <- c(1:20)
sizeInit <- 1
growSurv(sizeInit, sizeRange, data.frame(covariate=1), gr1, sv1)
plot(growSurv(sizeInit, sizeRange, data.frame(covariate=1), gr1, sv1),
    type="l", col = "dark gray",
xlab = "Continuous (e.g. size) stage at time t+1",
ylab = paste("Probability of survival to a specific size in t+1 from size ",
sizeInit, " at time t"))
```

---

growth *Estimates growth probabilities.*

Description

Generic function to predict the pdf (probability density function) of continuous (e.g. size) stage at t+1 given stage at t and a growth object.

Usage

```
growth(size, sizeNext, cov, growthObj)
```

Arguments

size	vector of current sizes.
sizeNext	vector of sizes in the next time-step.
cov	a data-frame with one row containing all covariates for this time-step.
growthObj	a growth object.

Details

Models defining continuous (size) stage at t+1, or growth increment, or log growth increment are defined; with various underlying statistical models allowing decreasing variance in size, etc.

Value

a vector of length sizeNext giving the pdf (probability density function) of each value of sizeNext.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Easterling, Ellner & Dixon. 2000. Size-specific sensitivity: a new structured population model. Ecology 81, p694-708.

See Also

`surv`, `growSurv`

Examples

```
dff <- generateData()
gr1 <- makeGrowthObj(dff)
sizeRange <- c(1:20)
sizeInit <- 1
growth(sizeInit, sizeRange, data.frame(cov=1), gr1)
plot(growth(sizeInit, sizeRange, data.frame(cov=1), gr1), type="l",
col = "dark gray", xlab = "Continuous (e.g. size) stage at time t+1",
ylab = paste("Probability of growth to a specific size in t+1 from size ",
sizeInit, " at time t"))
```

`growth-methods` ~~ *Methods for Function* `growth` ~~

Description

~~ Methods for function `growth` ~~

Methods

`signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObj")`
: Methods to predict the probability density of `sizeNext` via linear prediction based around a range of transforms of the current size, and covariates described in the data-frame "cov". The prediction supplies mean sizeNext directly; the density function around this is normal.

`signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjDecline`
: Methods to predict the probability density of `sizeNext` via linear prediction based around a range of transforms of the current size, and covariates described in the data-frame "cov". The prediction supplies mean sizeNext directly; the density function around this is normal and the sd may change with size.

`signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjHossfel`
: Methods to predict the probability density of `sizeNext` using the Hossfeld function. The prediction supplies mean incr which is added to size to obtain sizeNext; the density function around this is normal.

`signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjIncr")`
: Methods to predict the probability density of `sizeNext` via linear prediction based around a range of transforms of the current size, and covariates described in the data-frame "cov". The prediction supplies mean incr which is added to size to obtain sizeNext; the density function around this is normal.

`signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjIncrDec`
: Methods to predict the probability density of `sizeNext` via linear prediction based around a range of transforms of the current size, and covariates described in the data-frame "cov". The prediction supplies mean incr which is added to size to obtain sizeNext; the density function around this is normal and the sd may change with size.

`signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjLogIncr`
: Methods to predict the probability density of `sizeNext` via linear prediction based around a range of transforms of the current size, and covariates described in the data-frame "cov". The prediction supplies mean logincr which is exponentiated and added to size to obtain sizeNext; the density function around this is log normal; and variance changes with size.

`signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjLogIncr`
: Methods to predict the probability density of `sizeNext` via linear prediction based around a range of transforms of the current size, and covariates described in the data-frame "cov". The prediction supplies logincr which is exponentiated and added to size to obtain sizeNext; the density function around this is lognormal; the sd may change with size.

`signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjTruncIn`
: Methods to predict the probability density of `sizeNext` via linear prediction based around a range of transforms of the current size, and covariates described in the data-frame "cov". The prediction supplies mean incr which is added to size to obtain sizeNext; the density function around this is truncated normal, truncated at zero.

`signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjPois")`
: Methods to predict the probability density of `sizeNext` via linear prediction based around a range of transforms of the current size, and covariates described in the data-frame "cov". The prediction supplies mean sizeNext directly; the density function around this is poisson. NOTE THAT THIS EMPHATICALLY SHOULD NOT BE USED WITH AN IPM, SINCE THE PREDICTION IS NOT CONTINUOUS. It can be used with `makeIntegerPmatrix`.

`signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjNegBin"`
: Methods to predict the probability density of `sizeNext` via linear prediction based around a range of transforms of the current size, and covariates described in the data-frame "cov". The prediction supplies mean sizeNext directly; the density function around this is negative binomial. NOTE THAT THIS EMPHATICALLY SHOULD NOT BE USED WITH AN IPM, SINCE THE PREDICTION IS NOT CONTINUOUS. It can be used with `makeIntegerPmatrix`.

growthCum *Models growth allowing for cumulative bin estimation.*

Description

Generic function to predict the cdf (cummulative density function) of continuous (e.g. size) stage at t+1 given stage at t and a growth object.

Usage

```
growthCum(size, sizeNext, cov, growthObj)
```

Arguments

`size`	vector of current sizes.
`sizeNext`	vector of sizes in the next time step.
`cov`	covariate level in this time step (numeric of length 1).
`growthObj`	a growth object.

Value

a vector of length sizeNext giving the cdf (cummulative density function) of each value of sizeNext.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

See Also

growth

Examples

```
dff <- generateData()
gr1 <- makeGrowthObj(dff)
sizeRange <- c(1:20)
sizeInit <- 1
growthCum(sizeInit, sizeRange, data.frame(cov=1), gr1)
plot(growthCum(sizeInit, sizeRange, data.frame(cov=1), gr1), type="l",
col = "dark gray", xlab = "Continuous (e.g. size) stage at time t+1",
ylab = paste("Cummulative growth to a specific size in t+1 from size ",
sizeInit, " at time t"))
```

`growthCum-methods` *~~ Methods for Function* `growthCum` *~~*

Description

~~ Methods for function `growthCum` ~~

Methods

```
signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObj")

signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjDecline

signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjIncr")

signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjLogIncr

signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjTruncIn

signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjLogIncr

signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjHossfel
```

`growthModelComp` *Compares growth and survival objects built from different covariate sets.*

Description

Function compares model fits for growth and survival objects built with different linear combinations of covariates (plotting currently restricted to transforms of size; but comparison can include any chosen covariates). Growth can have multiple response forms. Returns a list containing a summary table of covariates and scores, and another list containing all of the growth (or survival) objects used in the comparison.

Usage

```
growthModelComp(dataf,
    expVars = c(sizeNext~1,  sizeNext~size, sizeNext~size + size2),
regressionType = "constantVar",  testType = "AIC",
makePlot = FALSE, mainTitle = "", plotLegend = TRUE,
  legendPos = "topright",...)
```

```
survModelComp(dataf, expVars = c(surv~1,  surv~size, surv~size + size2),
    testType = "AIC",
makePlot = FALSE, mainTitle = "",ncuts=20, plotLegend = TRUE,
   legendPos = "bottomleft", ...)
```

Arguments

`dataf`	dataframe with columns size, surv, and the growth response variable of choice
`expVars`	list of Formulas. Defaults to `c(sizeNext~1,  sizeNext~size, sizeNext~size + size2)`.
`regressionType`	character string identifying whether the type of regression run will have constant or changing variance (for `growthModelComp`. Defaults to `constantVar`.
`testType`	character string identifying the metric used to compare models. Can be any string that uses `loglike` from the lm or glm object. For example `"AIC"` or `"BIC"`. Defaults to `"AIC"`.
`makePlot`	logical whether to make plots with the comparison building. Defaults to `FALSE`.
`mainTitle`	string to place as the `main` attribute in plots (if `makePlot = TRUE`. defaults to `NULL`.
`ncuts`	for survModelComp, number of cuts in the data-set to be used in plotting
`plotLegend`	logical whether to plot a legend. Defaults to `FALSE`.
`legendPos`	places legend. Defaults to "topright".
`...`	additional arguments for plotting (ylim, col, etc)

Details

Both `growthModelComp` and `survModelComp` use a dataframe that has variables `size` and `sizeNext` to build a series of nested models. The default will build growth or survival objects with an intercept, an intercept and size, an an intercept with size and size^2 terms.

The models build use only `lm` or `glm` (and not `mcmcGLMM` for example) to estimate maximum likelihood estimates of functions. The testType (default `"AIC"` uses the `loglike` output from the lm or glm objects to score the model.

Plotting calls the functions `plotGrowthModelComp` or `plotSurvModelComp` to plot the objects. These functions can also be called after building the model comparison lists that are returned. If called outside of the initial building functions, they need to receive the `GrowthObjects` or `SurvObjects` list in the outputList from the build function. See `plotGrowthModelComp` and `plotSurvModelComp` for more details.

Value

a list with a summary table of class `dataframe` with models and scores and list of containing the objects of class grObj and survObj for each model.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

See Also

makeGrowthObj,makeSurvObj,plotGrowthModelComp, plotSurvModelComp

Examples

```
# Data with size and sizeNext
dff <- generateData()

growthModelComp(dff, makePlot = TRUE)
survModelComp(dff, makePlot = TRUE)

growthModelComp(dff, makePlot = TRUE, regressionType = "changingVar")
```

---

growthObj-class *Class "growthObj"*

Description

A class object description

Objects from the Class

Objects can be created by calls of the form `new("growthObj", ...)`.

Slots

`fit`: Object of class `"lm"` ~~

`sd`: Object of class `"numeric"` ~~

Methods

growth `signature(size = "numeric", sizeNext = "numeric", cov = "numeric", growthObj = "growthObj")`: ...

growthCum `signature(size = "numeric", sizeNext = "numeric",  cov = "numeric", growthObj = "growthObj`
...

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

Examples

```
showClass("growthObj")
```

`growthObjDeclineVar-class`
Class "growthObjDeclineVar"

Description

A class object description

Objects from the Class

Objects can be created by calls of the form `new("growthObjDeclineVar", ...)`.

Slots

`fit`**:** Object of class `"list"` ~~

Methods

growth `signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjDe`
...

growthCum `signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthO`
...

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

Examples

```
showClass("growthObjDeclineVar")
```

`growthObjHossfeld-class`
Class "growthObjHossfeld"

Description

A class object description

Objects from the Class

Objects can be created by calls of the form `new("growthObjHossfeld", ...)`.

Slots

`logLik`**:** Object of class `"numeric"` ~~

`paras`**:** Object of class `"numeric"` ~~

`sd`**:** Object of class `"numeric"` ~~

`hessian`**:** Object of class `"matrix"` ~~

Methods

growth `signature(size = "numeric", sizeNext = "numeric", cov = "numeric", growthObj = "growthObjHossf`
...

growthCum `signature(size = "numeric", sizeNext = "numeric", h = "numeric", cov = "numeric", growthOb`
...

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

Examples

```
showClass("growthObjHossfeld")
```

`growthObjIncr-class` *Class "growthObjIncr"*

Description

A class object description

Objects from the Class

Objects can be created by calls of the form `new("growthObjIncr", ...)`.

Slots

`fit`**:** Object of class `"lm"` ~~

`sd`**:** Object of class `"numeric"` ~~

Methods

growth `signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjIn`
...

growthCum `signature(size = "numeric", sizeNext = "numeric", h = "numeric", cov = "data.frame", growt`
...

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

Examples

```
showClass("growthObjIncr")
```

`growthObjIncrDeclineVar-class`
Class "growthObjIncrDeclineVar"

Description

A class object description

Objects from the Class

Objects can be created by calls of the form `new("growthObjIncrDeclineVar", ...)`.

Slots

`fit`: Object of class `"list"` ~~

Methods

growth `signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjIn`
...

growthCum `signature(size = "numeric", sizeNext = "numeric", h = "numeric", cov = "data.frame", growt`
...

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

Examples

```
showClass("growthObjIncrDeclineVar")
```

growthObjLogIncr-class *Class* "growthObjLogIncr"

Description

A class object description

Objects from the Class

Objects can be created by calls of the form `new("growthObjLogIncr", ...)`.

Slots

`fit`**:** Object of class `"lm"` ~~

`sd`**:** Object of class `"numeric"` ~~

Methods

growth `signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjLo`
...

growthCum `signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthO`
...

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

Examples

```
showClass("growthObjLogIncr")
```

growthObjLogIncrDeclineVar-class *Class* "growthObjLogIncrDeclineVar"

Description

A class object description

Objects from the Class

Objects can be created by calls of the form `new("growthObjLogIncrDeclineVar", ...)`.

Slots

`fit`**:** Object of class `"list"` ~~

Methods

growth `signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjLo`
...

growthCum `signature(size = "numeric", sizeNext = "numeric", h = "numeric", cov = "data.frame", growt`
...

Examples

```
showClass("growthObjLogIncrDeclineVar")
```

`growthObjNegBin-class` *Class "growthObjNegBin"*

Description

A class object description

Objects from the Class

Objects can be created by calls of the form `new("growthObjNegBin", ...)`.

Slots

`fit`**:** Object of class `"list"` ~~

Methods

growth `signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjNe`
...

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

Examples

```
showClass("growthObjNegBin")
```

`growthObjPois-class` *Class "growthObjPois"*

Description

A class object description

Objects from the Class

Objects can be created by calls of the form `new("growthObjPois", ...)`.

Slots

`fit`: Object of class `"list"` ~~

Methods

growth `signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjPo`
...

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

Examples

```
showClass("growthObjPois")
```

`growthObjTruncIncr-class` *Class* `"growthObjTruncIncr"`

Description

A class object description

Objects from the Class

Objects can be created by calls of the form `new("growthObjTruncIncr", ...)`.

Slots

`fit`: Object of class `"list"` ~~

`varcov`: Object of class `"matrix"` ~~

Methods

growth `signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthObjTr`
...

growthCum `signature(size = "numeric", sizeNext = "numeric", cov = "data.frame", growthObj = "growthO`
...

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

Examples

```
showClass("growthObjTruncIncr")
```

`Hossfeld` *Creates a Hossfeld function defining growth.*

Description

Functional form describing growth according to a Hossfeld function.

Usage

```
Hossfeld(size, par)
```

Arguments

`size`	vector of sizes.
`par`	vector of length 3.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Zuidema, Jongejans, Chien, During & Schieving. 2010. Integral projection models for trees: a new parameterization method and a validation of model output. Journal of Ecology 98, p345-355.

Rivas, Gonzalez, Gonzalez & von Gadow. 2004. Compatible height and site index models for five pine species in El Salto, Durango (Mexico). Forest Ecology and Management 201, p145-160.

Examples

```
dff <- generateData()
sizeRange <- c(1:20)
sizeInit <- 1
Hossfeld(sizeRange, rep(1, 3))
plot(Hossfeld(1:10, rep(1, 3)), type = "l",
    ylab = "Predicted increment from t to t+1",
xlab = "Continuous (size) stage in time t", col = "dark gray")
```

---

invLogit *Implements a logistic transform.*

Description

Provided a vector or numeric changes it into a vector on 0-1 using the invLogit transform.

Usage

```
invLogit(x)
```

Arguments

x	vector of numbers for which the transform is required.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

Examples

```
x <- rnorm(100)
plot(sort(x), invLogit(sort(x)), type = "l", xlab = "State x", ylab =
"Transformed state x", col = "dark gray")
```

IPMmatrix-class *Class* "IPMmatrix"

Description

Class IPMmatrix contains a matrix describing transitions between sizes or discrete stages; other slots described integration resolution, etc.

Objects from the Class

Objects can be created by calls of the form `new("IPMmatrix", ...)`.

Slots

`.Data`: Object of class `"matrix"` ~~

`nDiscrete`: Object of class `"numeric"` ~~

`nEnvClass`: Object of class `"numeric"` ~~

`nBigMatrix`: Object of class `"numeric"` ~~

`meshpoints`: Object of class `"numeric"` ~~

`env.index`: Object of class `"numeric"` ~~

`names.discrete`: Object of class `"character"` ~~

Extends

Class `"matrix"`, from data part. Class `"array"`, by class "matrix", distance 2. Class `"structure"`, by class "matrix", distance 3. Class `"vector"`, by class "matrix", distance 4, with explicit coerce.

Methods

No methods defined with class "IPMmatrix" in the signature.

Examples

```
showClass("IPMmatrix")
```

`IPMpackNews` *Show a text file containing package news and updates.*

Description

Learn about recent changes to IPMpack.

Usage

```
IPMpackNews()
```

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

Examples

```
IPMpackNews()
```

`largeMatrixCalc`	*Calculates population growth rate (lambda) and stable stage distribution in a computationally efficient way when the number of bins in the IPM is large.*

Description

Method to calculate population growth rate (lambda) and stable stage distribution where a large number of bins are used in the IPM, as it may be the case with species that vary largely in size, or models that include size x age interactions.

Usage

```
largeMatrixCalc(Pmatrix, Fmatrix, tol = 1e-08)
```

Arguments

`Pmatrix`	object of class IPMmatrix describing survival transitions.
`Fmatrix`	object of class IPMmatrix describing fecundity transitions.
`tol`	tolerance for convergence, defaults to 1e-08.

Value

`lambda`	Population rate of increase.
`stableDist`	Stable stage distribution.
`h1`	size bin width.

Note

Modified from Appendix A in Rees and Ellner 2009 (see references).

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Rees and Ellner. 2009. Integral projection models for populations in temporally varying environments. The American Naturalist 79, p575-594.

Caswell. 2001. Matrix population models: construction, analysis, and interpretation. 2nd ed. Sinauer. p377-502.

Garcia, Dahlgren, Ehrlen. 2011. No evidence of senescence in a 300-year-old mountain herb. Journal of Ecology 99, p1424-1430.

Examples

```
dff <- generateData()
Pmatrix <- makeIPMPmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE), growObj = makeGrowthObj(dff),
  survObj = makeSurvObj(dff))
Fmatrix <- makeIPMFmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE),fecObj = makeFecObj(dff))

largeMatrixCalc(Pmatrix, Fmatrix)

par(mfrow=c(1,2),pty="s")

plot(largeMatrixCalc(Pmatrix, Fmatrix)$stableDist,
    ylab = "Stable stage distribution",
xlab = "Continuous (e.g. size) stage",
type = "l", col = "blue", lty = 1, ylim = c(0:1))

#Note that this will not always run - as tolerance levels
# for convergence are set to be quite high
#plot(largeMatrixCalc(Pmatrix, Fmatrix)$reprodValue,
#    ylab = "Reproductive value",
# xlab = "Continuous (e.g. size) stage", type = "l", col = "red",
#    lty = 1, ylim = c(0:1))
```

`makeClonalObj` *Function to build clonal reproduction objects*

Description

Allows a series of different glms to be fit for all steps of clonal reproduction, e.g., probability of reproducing clonally, number of clonal offspring produced, etc. Currently only pre-census clonal reproduction relationships can be handled.

Usage

```
makeClonalObj(dataf, fecConstants=data.frame(NA),
Formula=list(fec~size),Family="gaussian",
Transform="none",meanOffspringSize=NA,
sdOffspringSize=NA,offspringSplitter=data.frame(continuous=1),
vitalRatesPerOffspringType=data.frame(NA),fecByDiscrete=data.frame(NA),
offspringSizeExplanatoryVariables="1", coeff=NULL,doOffspring=TRUE)
```

Arguments

`dataf`	a dataframe with columns "size", "sizeNext", "stage", "stageNext", and any number of columns with clonal reproduction data. If clonal reproduction data is transformed via log, etc, this MUST BE MADE CLEAR in the argument

`Transform` since the clonality object produced must generate total reproductive output.)

`fecConstants` a data-frame containing the value by which each of the product of the fecundity rates will be multiplied *in the order defined by the order supplied in the list* `Formula`; these might capture for example the probability of establishment of clones or other steps in the clonal reproductive pathway that are not measured for each parent; default is NA if no constants are used.

`Formula` a list containing formulas describing the desired explanatory variables (interactions, etc) and response variables in classical R style, i.e. covariates separated by '+', '*', ':'. Possible covariates include 'size', 'size2' (size^2), 'size3' (size^3),'logsize' (log(size)), and 'covariate' (if this name is used, the assumption is made that this is a discrete covariate from which compound matrices may be constructed), or any other covariates available in the data-frame supplied.

`Family` a character vector containing the names of the families to be used for the glms, e.g., binomial, poisson, etc. Again, these must appear *in the order defined by* `Formula`

`Transform` a character vector containing the names of the transforms to be used for the response variables, e.g., log, sqrt, -1, etc. Again, these must appear *in the order defined by* `Formula`

`meanOffspringSize`
numeric vector, defining mean offspring size. Defaults to NA, in which case the function will use to data to assess the mean offspring size according to the relationship defined in offspringSizeExplanatoryVariables (which either simply fits a mean, or may fit more complex relationships linking maternal size to offspring size).

`sdOffspringSize`
numeric vector, defining standard deviation of offspring size. Defaults to NA, in which case the function will use the data to assess the sd in offspring size; as described for meanOffspringSize

`offspringSplitter`
dataframe with values defining the number of offspring going into the indicated offspring category; will be re-scaled to sum to 1 within the function. This argument needs to be entered as a data.frame, and the names in the data.frame need to precisely match the used stage names in the data file.

`vitalRatesPerOffspringType`
dataframe defining which fecundity rates (both functions and constants) apply to which offspring category. This only needs to be specified when some fecundity rates do not apply to all offspring categories. The offspring categories in the column names of this dataframe should match those in the offspringSplitter exactly. The row names of the dataframe should match the fecundity column names in the data file and the supplied fecundity constants, in that order. In the dataframe, a '1' indicates that a fecundity rate applies to an offspring category, while a '0' indicates an omission. For instance, establishment and seedling survival rates may be applicable to seedlings, but not to seeds that go into a seedbank (depending on the life cycle and definition of vital rates).

`fecByDiscrete`	data.frame defining number of offspring produced by each discrete class ; defaults to 0. If specified, ALL discrete classes MUST appear in alphabetical order, so NO "continuous". e.g. fecByDiscrete=data.frame(dormant=0,seedAge1=4.2,seedOld=0)
`offspringSizeExplanatoryVariables`	a character defining the relationship defining offspring size; the default is "1", indicating simply fitting a mean and a variance; alternatives would including defining offspring size as a function of maternal size (i.e., offspringSizeExplanatoryVariables="size") or more complex polynomials of size (i.e., offspringSizeExplanatoryVariables="size+size2"). The corresponding relationship is fitted to the data contained in dataf, taking as the response variable the column "sizeNext" in dataf for rows where the column "offspringNext" is equal to "clonal" and the column "stageNext" is equal to "continuous".
`coeff`	list of numeric vector of required coefficients to be imposed if dataf is NULL for each element of the Formula list in order; must be compatible with Formula
`doOffspring`	argument defining whether you wish to fit an offspring regression as part of this function, or do it separately (see makeOffspringObj)

Details

This function fits a suite of subfunctions of clonal reproduction towards creating a Clonal transition projection model. Users can define the functional form of each relationship, as well as the distribution and any transforms. There is also a possibility of defining clonal reproduction from discrete sizes into each of the subfunction outcomes; defined in the matrix fecByDiscrete.

Value

an object of class fecObj

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

See Also

`makeSurvObj`, `makeGrowthObj`

Examples

```
# makeClonalObj works exactly the same way as makeFecObj.
# An example will be added here as soon as we have added a
# data file on a clonal plant to the package.
```

makeCompoundCmatrix *Builds a compound C matrix.*

Description

Uses clonality object, and environmental transition objects to construct a matrix defining probabilities for transitions between sizes due to clonal reporudction given both a continuous state (e.g. size) and environmental state, as well as a discrete stage if necessary. Currently only pre-census clonal reproduction can be handled. NOTE - old createCompoundCmatrix is being deprecated; use makeCompoundCmatrix instead.

Usage

```
makeCompoundCmatrix(nEnvClass = 2, nBigMatrix = 50,
minSize = -1, maxSize = 50, envMatrix, clonalObj,
integrateType ="midpoint", correction = "none",
preCensus = TRUE, survObj = NULL, growObj = NULL,
offspringObj = NULL)
```

Arguments

`nEnvClass`	numeric, number of environmental classes, defaults to 2.
`nBigMatrix`	numeric, number of size bins in the P matrix, defaults to 50.
`minSize`	numeric, minimum size of the P matrix, defaults to -1.
`maxSize`	numeric, maximum size of the P matrix, defaults to 50.
`envMatrix`	envMatrix object defining transitions between environmental states for each size.
`clonalObj`	clonality object.
`integrateType`	integration type, defaults to "midpoint" (which uses probability density function); other option is "cumul" (which uses the cumulative density function).
`correction`	correction type, defaults to `none`. The first option is `constant` which will multiply every column of the IPM by a constant sufficient to adjust values to those predicted for total fertility at that size. The second option is `discretizeExtremes` which will place all transitions to sizes smaller than `minSize` into the smallest bin, and transitions to sizes larger than `maxSize` into the largest bin.
`preCensus`	logical (TRUE or FALSE), indicating whether the fecundity object should represent an interval between pre-breeding or a post-breeding censusses. defaults to TRUE (pre-breeding census), meaning that all reproduction and offspring rates required for the F matrix are embedded in fecObj. Alternatively, an F matrix based on post-breeding census (preCensus=FALSE) requires additional survObj and growObj, to cover the survival and growth of the parents until the reproduction event.
`survObj`	suvival object, describing the survival of parents from a census until the reproduction event starts (at some point during the inter-census time step.

`growObj`	growth object, describing the growth of parents that survive until the reproduction event starts. Warning: this growth object is still ignored in makeIPMFmatrix in the current version of IPMpack. It will become functional in coming versions.
`offspringObj`	growth object, describing the size of offspring (this process may alternatively appear in fecObj).

Value

an object of class IPMmatrix with dimensions nBigMatrix*nEnvClass, or if discrete transitions exist (nBigMatrix+nDisc)*nEnvClass

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

For information on C matrix: Caswell. 2001. Matrix population models: construction, analysis, and interpretation. 2nd ed. Sinauer. p110-112.

For midpoint: Zuidema, Jongejans, Chien, During & Schieving. Integral projection models for trees: a new parameterization method and a validation of model output. Journal of Ecology 98, p345-355.

For multiple-vital rate integration on fecundity: Yang, Jongejans, Yang & Bishop. 2011. The effect of consumers and mutualists of Vaccinum membranaceum at Mount St. Helens: dependence on successional context. PLoS One 10, p1-11.

See Also

`makeCompoundPmatrix`,`makeIPMCmatrix`

Examples

```
## See makeCompoundFmatrix for examples
```

`makeCompoundFmatrix` *Builds a compound F matrix.*

Description

Uses fecundity object, and environmental transition objects to construct a matrix defining probabilities for transitions between sizes due to fecundity given both a continuous state (e.g. size) and environmental state, as well as a discrete stage if necessary (e.g. seedbank). NOTE - old createCompoundFmatrix is being deprecated; use makeCompoundFmatrix instead.

Usage

```
makeCompoundFmatrix(nEnvClass = 2, nBigMatrix = 50,
minSize = -1, maxSize = 50, envMatrix, fecObj, integrateType="midpoint",
correction="none",
preCensus = TRUE, survObj = NULL, growObj = NULL, offspringObj=NULL)
```

Arguments

`nEnvClass`	numeric, number of environmental classes, defaults to 2.
`nBigMatrix`	numeric, number of size bins in the P matrix, defaults to 50.
`minSize`	numeric, minimum size of the P matrix, defaults to -1.
`maxSize`	numeric, maximum size of the P matrix, defaults to 50.
`envMatrix`	envMatrix object defining transitions between environmental states for each size.
`fecObj`	fecundity object.
`integrateType`	integration type, defaults to "midpoint" (which uses probability density function); other option is "cumul" (which uses the cumulative density function).
`correction`	correction type, defaults to `none`. The first option is `constant` which will multiply every column of the IPM by a constant sufficient to adjust values to those predicted for total fertility at that size. The second option is `discretizeExtremes` which will place all transitions to sizes smaller than `minSize` into the smallest bin, and transitions to sizes larger than `maxSize` into the largest bin.
`preCensus`	logical (TRUE or FALSE), indicating whether the fecundity object should represent an interval between pre-breeding or a post-breeding censusses. defaults to TRUE (pre-breeding census), meaning that all reproduction and offspring rates required for the F matrix are embedded in fecObj. Alternatively, an F matrix based on post-breeding census (preCensus=FALSE) requires additional survObj and growObj, to cover the survival and growth of the parents until the reproduction event.
`survObj`	suvival object, describing the survival of parents from a census until the reproduction event starts (at some point during the inter-census time step.
`growObj`	growth object, describing the growth of parents that survive until the reproduction event starts. Warning: this growth object is still ignored in makeIPMFmatrix in the current version of IPMpack. It will become functional in coming versions.
`offspringObj`	growth object, describing the size of offspring (this process may alternatively appear in fecObj).

Value

an object of class IPMmatrix with dimensions nBigMatrix*nEnvClass, or if discrete transitions exist (nBigMatrix+nDisc)*nEnvClass

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

For information on F matrix: Caswell. 2001. Matrix population models: construction, analysis, and interpretation. 2nd ed. Sinauer. p110-112.

For midpoint: Zuidema, Jongejans, Chien, During & Schieving. Integral projection models for trees: a new parameterization method and a validation of model output. Journal of Ecology 98, p345-355.

For multiple-vital rate integration on fecundity: Yang, Jongejans, Yang & Bishop. 2011. The effect of consumers and mutualists of Vaccinum membranaceum at Mount St. Helens: dependence on successional context. PLoS One 10, p1-11.

See Also

`makeCompoundPmatrix`,`makeIPMFmatrix`

Examples

```
# Data with only continuous stage and two habitats
dff <- generateData()
dff$fec[dff$fec==0] <- NA
Fmatrix <-makeCompoundFmatrix(nBigMatrix = 20,
minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm=TRUE),
envMatrix = makeEnvObj(dff),
fecObj = makeFecObj(dff, Formula = fec~size+size2+covariate,
Transform = "log"))

image(1:nrow(Fmatrix), 1:ncol(Fmatrix), t(log(Fmatrix)),
xlab="Continuous state (e.g. size) at t",
ylab="Continuous state (e.g. size) at t+1", axes = FALSE)
axis(1, at = 1:nrow(Fmatrix), lab = round(rep(Fmatrix@meshpoints,
    Fmatrix@nEnvClass), 2))
axis(2,at = 1:nrow(Fmatrix), lab = round(rep(Fmatrix@meshpoints,
    Fmatrix@nEnvClass), 2))
abline(h = length(Fmatrix@meshpoints)*(1:Fmatrix@nEnvClass))
abline(v = length(Fmatrix@meshpoints)*(1:Fmatrix@nEnvClass))

# Data with continuous and discrete stages
dff <- generateData(type="discrete")
dff$fec[dff$fec==0] <- NA
dff$covariate <- sample(1:3, size = nrow(dff), replace = TRUE)
dff$covariateNext <- sample(1:3, size = nrow(dff), replace = TRUE)
fv1 <- makeFecObj(dff, Formula = fec~size, Transform = "log",
offspringSplitter=data.frame(continuous = 0.9, dormant = 0.1))
Fmatrix <- makeCompoundFmatrix(minSize = min(dff$size, na.rm=TRUE),
maxSize = max(dff$size, na.rm = TRUE), envMatrix = makeEnvObj(dff),
    fecObj = fv1)
```

makeCompoundPmatrix *Builds a compound P matrix.*

Description

Uses growth, survival, discreteTrans, and environmental transition objects to construct a matrix defining probabilities for transitions between continuous stages (e.g. size) due to growth and survival given both size and environmental state and discrete stages. NOTE - old createCompoundPmatrix is being deprecated; use makeCompoundPmatrix instead.

Usage

```
makeCompoundPmatrix(nEnvClass = 2, nBigMatrix = 50,
minSize = -1, maxSize = 50, envMatrix, growObj, survObj,
discreteTrans = 1, integrateType = "midpoint",correction = "none")
```

Arguments

`nEnvClass`	numeric, number of environmental classes, defaults to 2.
`nBigMatrix`	numeric, number of size bins in the P matrix, defaults to 50.
`minSize`	numeric, minimum size of the P matrix, defaults to -1.
`maxSize`	numeric, maximum size of the P matrix, defaults to 50.
`envMatrix`	envMatrix object defining transitions between environmental states for each size.
`growObj`	growth object.
`survObj`	survival object.
`discreteTrans`	object of class discreteTrans, or numeric.
`integrateType`	integration type, defaults to "midpoint" (which uses probability density function); other option is "cumul" (which uses the cumulative density function).
`correction`	correction type, defaults to `none`. The first option is `constant` which will multiply every column of the IPM by a constant sufficient to adjust values to those predicted for survival at that size. The second option is `discretizeExtremes` which will place all transitions to sizes smaller than `minSize` into the smallest bin, and transitions to sizes larger than `maxSize` into the largest bin.

Details

This structure can also be used to define size x age IPMs, where the transition between ages is reflected by a similar matrix.

Value

an object of class IPMmatrix with dimensions nBigMatrix * nEnvClass, or if discrete transitions exist (nBigMatrix + nDisc) * nEnvClass

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

For information on P matrix: Caswell. 2001. Matrix population models: construction, analysis, and interpretation. 2nd ed. Sinauer. p110-112.

For habitat x stage modeling: Tuljapurkar, Horvitz & Pascarella. 2003. The many growth rates and elasticities of populations in random environments. American Naturalist 162: p489-502.

Pascarella & Horvitz. 1998. Hurricane disturbance and the population dynamics of a tropical understory shrub: megamatrix elasticity analysis. Ecology 79: p547-563.

Horvitz & Schemske. 1986. Seed dispersal and environmental heterogeneity in a neotropical herb: A model of population and patch dynamics. In Symposium on frugivores and seed dispersal . (Estrada & Fleming, eds.) Dr. W. Junk Publishers, Dordrecht, Netherlands. pp. 169-186.

For age x size modeling: Garcia, Dahlgren and Ehrlen. 2011. No evidence of senescence in a 300-year-old mountain herb. Journal of Ecology 99, p1424-1430.

For general information:

Easterling, Ellner and Dixon. 2000. Size-specific sensitivity: a new structured population model. Ecology 81, p694-708.

Ellner and Rees. 2006. Integral projection models for species with complex demography. The American Naturalist 167, p410-428.

See Also

makeCompoundFmatrix,makeIPMPmatrix

Examples

```
# Data with only continuous stage and two habitats
dff <- generateData()
Pmatrix <- makeCompoundPmatrix(minSize = min(dff$size,na.rm = TRUE),
maxSize = max(dff$size,na.rm = TRUE), envMatrix = makeEnvObj(dff),
growObj = makeGrowthObj(dff, Formula = sizeNext~size+size2+covariate),
survObj = makeSurvObj(dff, Formula = surv~size+size2+covariate))

image(1:nrow(Pmatrix), 1:ncol(Pmatrix), t(log(Pmatrix)),
xlab = "Continuous stage (e.g. size) at t",
ylab = "Continuous stage (e.g. size) at t+1", axes = FALSE)
axis(1, at = 1:nrow(Pmatrix), lab = round(rep(Pmatrix@meshpoints,
    Pmatrix@nEnvClass), 2))
axis(2, at = 1:nrow(Pmatrix), lab = round(rep(Pmatrix@meshpoints,
    Pmatrix@nEnvClass), 2))
abline(h = length(Pmatrix@meshpoints) * (1:Pmatrix@nEnvClass))
abline(v = length(Pmatrix@meshpoints) * (1:Pmatrix@nEnvClass))

# Data with continuous and discrete stages
dff <- generateData(type="discrete")
```

```
dff$covariate <- sample(1:3, size = nrow(dff), replace = TRUE)
dff$covariateNext <- sample(1:3, size = nrow(dff), replace = TRUE)
discM <- makeDiscreteTrans(dff)
Pmatrix <- makeCompoundPmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE), envMatrix = makeEnvObj(dff),
growObj = makeGrowthObj(dff, Formula = sizeNext~size+size2+covariate),
survObj = makeSurvObj(dff, Formula = surv~size+size2+covariate),
discreteTrans = discM)
```

---

makeDiscreteTrans — *Builds matrix for transitions between discrete (e.g. seedbank, dormant) and continuous (e.g. size) stages.*

Description

Function that takes a data frame reflecting a population that includes continuous (e.g., size) and discrete (e.g., diapause) classes of individuals, and builds a transition matrix from this for movement between discrete and continuous stages (providing just a single value for the continuous stages; the detail of movement between continuous stages (e.g., sizes) is generated elsewhere). This object can then be used as an argument in the function to "makeIPMPmatrix" to build a P matrix that contains both discrete and continuous stages.

The dataframe must contain columns "stage" and "stageNext" with values of the names of the discrete classes and the name "continuous" where appropriate, in the current and subsequent time step. Continuous categories are also defined by their measurements, contained in "size" and "sizeNext". This is necessary for defining the mean and variance of the continuous stage of individuals emerging from discrete stages. Alternatively, you can enter the transition matrix for the discrete stages in the 'discreteTrans' argument.

Usage

```
makeDiscreteTrans(dataf, stages = NA, discreteTrans = NA,
    meanToCont = NA, sdToCont = NA,
        continuousToDiscreteExplanatoryVariables = "size")
```

Arguments

`dataf`	a dataframe with columns "stage", "stageNext" (both FACTORS (use `as.factor`) containing either names of discrete stages or "continuous" or "dead"), "size", "sizeNext", and "surv" (continuous and boolean variables respectively).
`stages`	a character vector with the names of the discrete classes. Normally this argument does not have to be used as the names of discrete classes are extracted from the data or entered discreteTrans matrix.
`discreteTrans`	a matrix with transition probabilities between the discrete and continuous stages. The column names should match the stage/stageNext names in the data file. The names discrete stages should be in alphabetical order, followed by 'continuous'. The row names should be the same, but with 'dead' added at the bottom. Thus, this matrix represents all the fates of individuals from the various classes.

`meanToCont` a matrix containing the mean sizeNext values for individuals moving from discrete classes to the continuous classes (should contain NA when no individuals move from a particular discrete class to a continuous sizeNext). The column names should be equal (in alphabetical order) to the names of the discrete classes (so no continuous).

`sdToCont` a matrix containing the sd sizeNext values for individuals moving from discrete classes to the continuous classes (should contain NA when no individuals move from a particular discrete class to a continuous sizeNext). The column names should be equal (in alphabetical order) to the names of the discrete classes (so no continuous).

`continuousToDiscreteExplanatoryVariables`
a character defining the relationship defining size influences the probability of individuals in the continuous class moving to any of the discrete classes. This argument is not relevant when individuals in the continuous stage cannot move into any discrete stage.

Value

an object of class "discreteTrans" with columns corresponding to all the discrete and the one continuous stage.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Easterling, Ellner & Dixon. 2000. Size-specific sensitivity: a new structured population model. Ecology 81, p694-708.

Ellner & Rees. 2006. Integral projection models for species with complex demography. The American Naturalist 167, p410-428.

See Also

`makeIPMPmatrix`

Examples

```
dff <- generateData(type="discrete")
discTrans <- makeDiscreteTrans(dff)
makeIPMPmatrix(nBigMatrix = 10, growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff), discreteTrans = discTrans)

# Often some data needed for makeDiscreteTrans are not available as
# individual records in your data file. For instance the fate of seeds
# in the seedbank of the generateData(type="discrete") example:
dff<-generateData()
# Now the transition matrix needs to be entered as an argument,
# as well as the mean and sd of sizeNext values for discrete to
# continuous transitions.
discTrans <- makeDiscreteTrans(dff,
discreteTrans = matrix(c(.5,.1,.4,0,1,0),
nrow=3, ncol=2,
dimnames = list(c("seedbank","continuous","dead"),
c("seedbank","continuous"))),
meanToCont = matrix(c(mean(dff$sizeNext[is.na(dff$stage)
    &dff$stageNext=="continuous"],na.rm=TRUE)),
nrow=1, ncol=1, dimnames = list(1,c("seedbank"))),
sdToCont = matrix(c(sd(dff$sizeNext[is.na(dff$stage)
  &dff$stageNext=="continuous"],na.rm=TRUE)),
nrow=1, ncol=1, dimnames = list(1,c("seedbank"))))
Pmatrix<-makeIPMPmatrix(nBigMatrix = 10, growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff), discreteTrans = discTrans)
Fmatrix<-makeIPMFmatrix(nBigMatrix = 10, fecObj = makeFecObj(dff))
dim(Pmatrix)
dim(Fmatrix)
# To fix this mismatch, all discrete classes (1 in this case) need to
# be added to the Fmatrix as well, e.g.:
Fmatrix<-makeIPMFmatrix(nBigMatrix = 10, fecObj = makeFecObj(dff,
    offspringSplitter=data.frame(seedbank=.3,continuous=.7)))
IPM <- Pmatrix + Fmatrix
```

makeDiscreteTransInteger

Builds matrix for transitions between discrete (e.g. seedbank, dormant) and continuous (e.g. size) stages, for situation where continuous variable is an integer (e.g. number of leaves).

Description

Function that takes a data frame reflecting a population that includes continuous (e.g., size) and discrete (e.g., diapause) classes of individuals, and builds a transition matrix from this for movement between discrete and continuous stages (providing just a single value for the continuous stages; the detail of movement between continuous stages (e.g., sizes) is generated elsewhere). This object can then be used as an argument in the function to "makeIPMPmatrix" to build a P matrix that contains both discrete and continuous stages.

The dataframe must contain columns "stage" and "stageNext" with values of the names of the discrete classes and the name "continuous" where appropriate, in the current and subsequent time step. Continuous categories are also defined by their measurements, contained in "size" and "sizeNext". This is necessary for defining the mean and variance of the continuous stage of individuals emerging from discrete stages. Alternatively, you can enter the transition matrix for the discrete stages in the 'discreteTrans' argument.

Usage

```
makeDiscreteTransInteger(dataf,
stages = NA,
discreteTrans = NA,
meanToCont = NA,
thetaToCont = NA,
continuousToDiscreteExplanatoryVariables = "size",
distToCont = "poisson")
```

Arguments

`dataf`	a dataframe with columns "stage", "stageNext" (both FACTORS (use `as.factor`) containing either names of discrete stages or "continuous" or "dead"), "size", "sizeNext", and "surv" (continuous and boolean variables respectively).
`stages`	a character vector with the names of the discrete classes. Normally this argument does not have to be used as the names of discrete classes are extracted from the data or entered discreteTrans matrix.
`discreteTrans`	a matrix with transition probabilities between the discrete and continuous stages. The column names should match the stage/stageNext names in the data file. The names discrete stages should be in alphabetical order, followed by 'continuous'. The row names should be the same, but with 'dead' added at the bottom. Thus, this matrix represents all the fates of individuals from the various classes.
`meanToCont`	a matrix containing the mean sizeNext values for individuals moving from discrete classes to the continuous classes (should contain NA when no individuals move from a particular discrete class to a continuous sizeNext). The column names should be equal (in alphabetical order) to the names of the discrete classes (so no continuous).
`thetaToCont`	a matrix containing the size parameter of sizeNext values for individuals moving from discrete classes to the continuous classes (should contain NA when no individuals move from a particular discrete class to a continuous sizeNext). The column names should be equal (in alphabetical order) to the names of the discrete classes (so no continuous).
`continuousToDiscreteExplanatoryVariables`	a character defining the relationship defining size influences the probability of individuals in the continuous class moving to any of the discrete classes. This argument is not relevant when individuals in the continuous stage cannot move into any discrete stage.
`distToCont`	character indicating the desired distribution of emergent sizes (poisson or negative binomial)

Details

See help for makeDiscreteTrans; this is exactly analagous, except that it uses the poisson or negative binomial as descriptors for sizes on emergence from discrete stages

Value

an object of class "discreteTransInteger" with columns corresponding to all the discrete and the one continuous stage.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Easterling, Ellner & Dixon. 2000. Size-specific sensitivity: a new structured population model. Ecology 81, p694-708.

Ellner & Rees. 2006. Integral projection models for species with complex demography. The American Naturalist 167, p410-428.

See Also

`makeIPMPmatrix`, `makeDiscreteTrans`

Examples

```
dff <- generateData(type="discrete")
dff$fec[dff$fec==0] <- NA
dff$size <- pmax(floor(dff$size+10),0)
dff$sizeNext <- pmax(floor(dff$sizeNext+10),0)
discTrans <- makeDiscreteTransInteger(dff)
makeIPMPmatrix(nBigMatrix = 10, growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff), discreteTrans = discTrans)
```

`makeEnvObj` *Builds environmental transition objects.*

Description

Function that takes vectors of discrete environmental states (e.g. shaded vs open canopy habitats) at time t and time t+1 and builds a transition (habitat) matrix from these.

Usage

```
makeEnvObj(dataf)
```

Arguments

`dataf`	a dataframe with columns 'covariate' and 'covariatel' indicating environmental covariate values at times t, and at t+1, respectively; these must take values of sequential integers, starting at'1'.

Value

an object of class envMatrix.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Horvitz & Schemske. 1995. Spatiotemporal variation in demographic transitions of a tropical understory herb. Projection matrix analysis. Ecological Monographs 65, p155-192.

Horvitz, Ehrlen & Matlaga. 2010. Context-dependent pollinator limitation in stochastic environments: can increased seed set overpower cost of reproduction in an understorey herb? Journal of Ecology 98, p268-278.

See Also

`makeFecObj`, `makeSurvObj`, `makeIPMPmatrix`, `makeIPMFmatrix`

Examples

```
dff <- generateData()
env <- makeEnvObj(dff)
env
```

---

makeFecObj   *Function to build fecundity objects*

Description

Allows a series of different glms to be fit all on the way to fecundity, e.g., probability of flowering, number of flower heads produced, etc; as well as fecundity into different discrete classes. Alternatively, the function creates fecundity objects following a specified list of formulas with specified corresponding list of coefficients.

Usage

```
makeFecObj(dataf, fecConstants=data.frame(NA),
Formula=list(fec~size),Family="gaussian",
Transform="none",meanOffspringSize=NA,
sdOffspringSize=NA,offspringSplitter=data.frame(continuous=1),
vitalRatesPerOffspringType=data.frame(NA),fecByDiscrete=data.frame(NA),
offspringSizeExplanatoryVariables="1", coeff=NULL,doOffspring=TRUE,
reproductionType="sexual")
```

Arguments

`dataf` a dataframe with columns "size", "sizeNext", "stage", "stageNext", and any additional columns with fecundity data. UIf fecundity data is transformed via log, etc, this MUST BE MADE CLEAR in the argument `Transform` since the fecundity object produced must generate total reproductive output.)

`fecConstants` a list containing the value by which each of the fecundity rates will be multiplied *in the order defined by the order in* `Formula`. This data frame adjusts the probability of establishment of seeds or other stages in sexual reproduction that are not explicitly incorporated for each parent (e.g., 25% of seeds across all individuals germinate). The default is NA if no constants are used (equivalent to multiplying by 1).

`Formula` a formulas describing the desired explanatory variables (interactions, etc) in classical R style, i.e. separated by '+', '*', ':' and the response variables of choice. Possible covariates include 'size', 'size2' (size^2), 'size3' (size^3),'logsize' (log(size)), and 'covariate' (if this name is used, the assumption is made that this is a discrete covariate from which compound matrices may be constructed), and any other covariates available in dataf. Again, these must appear *in the order defined by the Formula argument*. See `formula` in `base` for more details.

`Family` a character vector containing the names of the families to be used for the glms, e.g., binomial, poisson, etc. Again, these must appear *in the order defined by* `Formula`

`Transform` a character vector containing the names of the transforms to be used for the response variables, e.g., log, sqrt, -1, etc. Again, these must appear *in the order defined by* `Formula`

`meanOffspringSize`
numeric vector, defining mean offspring size. Defaults to NA, in which case the function will use to data to assess the mean offspring size according to the relationship defined in offspringSizeExplanatoryVariables (which either simply fits a mean, or may fit more complex relationships linking maternal size to offspring size).

`sdOffspringSize`
numeric vector, defining standard deviation of offspring size. Defaults to NA, in which case the function will use the data to assess the standard deviation of offspring size; as described for meanOffspringSize

`offspringSplitter`
dataframe with values defining the number of offspring going into the indicated offspring category; will be re-scaled to sum to 1 within the function. This argument needs to be entered as a data.frame, and the names in the data.frame need to precisely match the used stage names in the data file.

`vitalRatesPerOffspringType`
dataframe defining which fecundity rates (both functions and constants) apply to which offspring category. This only needs to be specified when some fecundity rates do not apply to all offspring categories. The offspring categories in the column names of this dataframe should match those in the offspringSplitter exactly. The row names of the dataframe should match the fecundity column names in the data file and the supplied fecundity constants, in that order. In the dataframe,

a '1' indicates that a fecundity rate applies to an offspring category, while a '0' indicates an omission. For instance, establishment and seedling survival rates may be applicable to seedlings, but not to seeds that go into a seedbank (depending on the life cycle and definition of vital rates).

`fecByDiscrete`	data.frame defining number of offspring produced by each discrete class ; defaults to 0. If specified, ALL discrete classes MUST appear in alphabetical order, so NO "continuous". e.g. fecByDiscrete=data.frame(dormant=0,seedAge1=4.2,seedOld=0)
`offspringSizeExplanatoryVariables`	a character defining the relationship defining offspring size; the default is "1", indicating simply fitting a mean and a variance; alternatives would including defining offspring size as a function of maternal size (i.e., offspringSizeExplanatoryVariables="size") or more complex polynomials of size (i.e., offspringSizeExplanatoryVariables="size+size2"). The corresponding relationship is fitted to the data contained in dataf, taking as the response variable the column "sizeNext" in dataf for rows where the column "offspringNext" is equal to "sexual" and the column "stageNext" is equal to "continuous".
`coeff`	list of numeric vector of required coefficients to be imposed if dataf is NULL for each element of the Formula list in order; must be compatible with Formula
`doOffspring`	argument defining whether you wish to fit an offspring regression as part of this function (TRUE), or do it separately (see makeOffspringObj); if this is FALSE, other argument relating to the offspring distribution will be ignored
`reproductionType`	argument specifying if "sexual" or "clonal" offspring should be considered; default is "sexual".

Details

This function fits a suite of subfunctions of fecundity towards creating a Fecundity transition projection model; e.g., the probability of flowering as a function of size, the number of seeds produced as a function of size, etc. Users can define the functional form of each relationship, as well as the distribution and any transforms. There is also a possibility of defining reproduction from discrete sizes into each of the subfunction outcomes; defined in the matrix fecByDiscrete.

Note that it is crucial that users appropriately set up the data to adequately reflect conditionality in the fertility kernel; for example, if there are two columns, with one reflecting the probability of flowering (0s and 1s) and the other reflecting seed output (integers) it is important that where the probability of flowering is 0, seed output is set to NA, as otherwise, meaningless 0s in the seed output column will bias the regression.

Finally, if the same variables are used in multiple equations, users should check for problems of endogeneity.

Value

an object of class fecObj

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

See Also

makeSurvObj, makeGrowthObj, makeOffspringObj

Examples

```
dff <- generateData(type="discrete")
fv1 <- makeFecObj(dff,
offspringSplitter = data.frame(continuous = 0.2, seedAge1 = 0.8,
seedOld = 0),
fecByDiscrete = data.frame(dormant = 0, seedAge1 = 4.2, seedOld = 0))

#now without any data, imposing desired coefficients
fv1 <- makeFecObj(Formula=list(Fec~size),
offspringSplitter = data.frame(continuous = 0.2, dormant=0,
seedAge1 = 0.8),
fecByDiscrete = data.frame(dormant = 0, seedAge1 = 4.2),
coeff=list(c(1,1)), meanOffspringSize=1,sdOffspringSize=1)
```

---

makeFecObjInteger *Function to build fecundity objects*

Description

Allows a series of different glms to be fit all on the way to fecundity, e.g., probability of reproducing, number of reproductive structures produced (e.g. flowers), etc; as well as fecundity into different discrete classes.

Usage

```
makeFecObjInteger(dataf,
fecConstants = data.frame(NA),
Formula = list(fec~size),
Family = "gaussian",
Transform = "none",
meanOffspringSize = NA,
thetaOffspringSize = NA,
offspringSplitter = data.frame(continuous=1),
vitalRatesPerOffspringType=data.frame(NA),
fecByDiscrete = data.frame(NA),
offspringSizeExplanatoryVariables = "1",
distOffspring = "poisson",
coeff=NULL, doOffspring=TRUE,
reproductionType="sexual")

makeClonalObjInteger(dataf,
fecConstants = data.frame(NA),
Formula = list(fec~size),
Family = "gaussian",
Transform = "none",
meanOffspringSize = NA,
thetaOffspringSize = NA,
offspringSplitter = data.frame(continuous=1),
vitalRatesPerOffspringType = data.frame(NA),
fecByDiscrete = data.frame(NA),
offspringSizeExplanatoryVariables = "1",
distOffspring = "poisson",
coeff=NULL, doOffspring=TRUE)
```

Arguments

`dataf`	a dataframe with columns "size", "sizeNext", "stage", "stageNext", and any additional columns with fecundity data. If fecundity data is transformed via log, etc, this MUST BE MADE CLEAR in the argument `Transform` since the fecundity object produced must generate total reproductive output.)
`fecConstants`	a list containing the value by which each of the fecundity rates will be multiplied *in the order defined by the order in* `Formula`. This data frame adjusts the probability of establishment of seeds or other stages in sexual reproduction that are not explicitly incorporated for each parent (e.g., 25% of seeds across all individuals germinate). The default is NA if no constants are used (equivalent to multiplying by 1).
`Formula`	a formulas describing the desired explanatory variables (interactions, etc) in classical R style, i.e. separated by '+', '*', ':' and the response variables of choice. Possible covariates include 'size', 'size2' (size^2), 'size3' (size^3),'logsize' (log(size)), and 'covariate' (if this name is used, the assumption is made that this is a discrete covariate from which compound matrices may be constructed), and any other covariates available in dataf. Again, these must appear *in the order defined by the Formula argument*. See `formula` in base for more details.
`Family`	a character vector containing the names of the families to be used for the glms, e.g., binomial, poisson, etc. Again, these must appear *in the order defined by* `Formula`
`Transform`	a character vector containing the names of the transforms to be used for the response variables, e.g., log, sqrt, -1, etc. Again, these must appear *in the order defined by* `Formula`
`meanOffspringSize`	numeric vector, defining mean offspring size. Defaults to NA, in which case the function will use to data to assess the mean offspring size according to the relationship defined in offspringSizeExplanatoryVariables (which either simply fits a mean, or may fit more complex relationships linking maternal size to offspring size).
`thetaOffspringSize`	numeric vector, defining size parameter of offspring size. It is only required if the family of offspring size is negative binomial (rather than poisson). Defaults

	to NA, in which case the function will use the data to assess this parameter using dbinom.
`offspringSplitter`	dataframe with values defining the number of offspring going into the indicated offspring category; will be re-scaled to sum to 1 within the function. This argument needs to be entered as a data.frame, and the names in the data.frame need to precisely match the used stage names in the data file.
`vitalRatesPerOffspringType`	dataframe defining which fecundity rates (both functions and constants) apply to which offspring category. This only needs to be specified when some fecundity rates do not apply to all offspring categories. The offspring categories in the column names of this dataframe should match those in the offspringSplitter exactly. The row names of the dataframe should match the fecundity column names in the data file and the supplied fecundity constants, in that order. In the dataframe, a '1' indicates that a fecundity rate applies to an offspring category, while a '0' indicates an omission. For instance, establishment and seedling survival rates may be applicable to seedlings, but not to seeds that go into a seedbank (depending on the life cycle and definition of vital rates).
`fecByDiscrete`	data.frame defining number of offspring produced by each discrete class ; defaults to 0. If specified, ALL discrete classes MUST appear in alphabetical order, so NO "continuous". e.g. fecByDiscrete=data.frame(dormant=0,seedAge1=4.2,seedOld=0)
`offspringSizeExplanatoryVariables`	a character defining the relationship defining offspring size; the default is "1", indicating simply fitting a mean and a variance; alternatives would including defining offspring size as a function of maternal size (i.e., offspringSizeExplanatoryVariables="size") or more complex polynomials of size (i.e., offspringSizeExplanatoryVariables="size+size2"). The corresponding relationship is fitted to the data contained in dataf, taking as the response variable the column "sizeNext" in dataf for rows where the column "offspringNext" is equal to "sexual" and the column "stageNext" is equal to "continuous".
`distOffspring`	character indicating the desired distribution of offspring sizes (poisson or negative binomial)
`coeff`	not yet implemented
`doOffspring`	argument defining whether you wish to fit an offspring regression as part of this function (TRUE), or do it separately (see makeOffspringObj); if this is FALSE, other argument relating to the offspring distribution will be ignored
`reproductionType`	argument specifying if "sexual" or "clonal" offspring should be considered; default is "sexual".

Details

See help for makeFecObj; this is exactly analagous, except that it uses the poisson or negative binomial as descriptors for offspring size

Value

an object of class fecObjInteger

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

See Also

`makeSurvObj`, `makeGrowthObj`,`makeFecObj`

Examples

```
# Open dataset for the herbaceous perennial Cryptantha flava where
# the state variable is integer (number of rosettes)
data(dataIPMpackCryptantha)
head(dataIPMpackCryptantha)
d <- dataIPMpackCryptantha

#See the description of the data for information on the variables
help(dataIPMpackCryptantha)

# For this example, focus only on the first annual transition
# available in the dataset
d1 <- d[d$year==2004, ]

#Create fecundity object with integer data
fo <- makeFecObjInteger(d1, Formula = fec1~size, distOffspring = "poisson")


#Example with imposed offspring object
off1 <- makeOffspringObj(dataf = d1, Formula = sizeNext~1, Family="poisson")
fo <- makeFecObjInteger(d1, Formula = fec1~size, distOffspring = "poisson",
doOffspring=FALSE)
Fmatrix <-  makeIntegerFmatrix(fecObj = fo, offspringObj=off1)
```

`makeGrowthObj` *Function to build growth objects*

Description

A function that fits regressions that define growth (following next size, size increment, or log size increment) and from these build growth objects for which methods to build an IPM object are defined; alternatively, the function creates growth objects following a specified formula with specified coefficients and sd.

Usage

```
makeGrowthObj(dataf=NULL, Formula = sizeNext ~ size,
regType = "constantVar", Family = "gaussian", link = NULL, coeff=NULL,sd=NULL)
```

Arguments

`dataf`	a dataframe with columns ‘size’ and ‘sizeNext’(‘size’ is size at t, ‘sizeNext’ is size at t+1); facultatively, dataf may include ‘covariate’ and ‘covariatel’ for a single discrete covariate, indicating values at t, and at t+1, respectively; these must take values of sequential integers, starting at ‘1’. For models fitting growth increment, ’incr’ or ’logincr’ may be directly provided as a column in the dataframe, otherwise they are calculated as dataf$sizeNext - dataf$size or log(dataf$sizeNext - dataf$size), respectively.
`Formula`	a formula describing the desired explanatory variables (interactions, etc) according to the R notation for `formula`. style, i.e. separated by ‘+’, ‘*’, ‘:’ and response variable. Possible covariates include ‘size’, ’size2’ (size^2), ‘size3’ (size^3),‘logsize’ (log(size)), ‘logsize2’ (log(size)^2), and ‘covariate’ (if this name is used, the assumption is made that this is a discrete covariate from which compound matrices may be constructed); or any other covariate available in dataf.
`regType`	possible values include ‘constantVar’ or ‘changingVar’
`Family`	possible values include ‘gaussian’, ‘poisson’, ‘negbin’
`link`	defaults to NULL, currently only relevant with Family="negbin", and only "log" and "identity" are permitted
`coeff`	numeric vector of required coefficients to be imposed if dataf is NULL; must be compatible with Formula
`sd`	numeric of required sd to be imposed if dataf is NULL

Value

An object of class growthObj, or growthObjPois, growthObjIncr, or growthObjLogIncr; or growthObjDeclineVar, or growthObjIncrDeclineVar, or growthObjLogIncrDeclineVar. These are S4 objects which contain the slots:

`fit`	an object of class `lm` or `glm` or `gls` that can be used with predict in the growth methods

Slots can be listed by using slotNames(growthObj)

Note

See manual for details on building case-specific growth objects. Note that DeclineVar objects cannot currently be constructed without a data-frame.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez & Eelke Jongejans

See Also

makeSurvObj, makeFecObj

Examples

```
#generate data
dff <- generateData()
#make simple linear regression growth object relating size to size at t+1
(gr1 <- makeGrowthObj(dataf = dff, Formula = sizeNext~size))
#same but relating size to incr
(gr1 <- makeGrowthObj(dataf = dff, Formula = incr ~ size))

#assess fit
picGrow(dff,gr1)

#same but relating size to incr and discrete covariate
(gr1 <- makeGrowthObj(dataf = dff, Formula = incr ~ size + covariate,
regType = "changingVar"))
#with declining increment
(gr1 <- makeGrowthObj(dataf = dff, Formula = incr ~ size + covariate,
regType = "changingVar"))

#now specifying parameters and supplying no data
gr1 <- makeGrowthObj(Formula = incr ~ size + covariate,coeff=c(1,1,1),sd=1)
```

---

makegrowthObjHossfeld *Function to make a Hossfeld Growth Object*

Description

Takes a data-frame with at minimum columns size and sizeNext; and fits a Hossfeld type growth function to increment - if length(dataf$incr) is zero, it will calculate dataf$increment as the difference between size and sizeNext; otherwise it will take the column provided

Usage

```
makegrowthObjHossfeld(dataf)
```

Arguments

dataf	a dataframe with columns 'size' and 'sizeNext'('size' is size at t, 'sizeNext' is size at t+1);

Value

Returns a growth object of class growthObjHossfeld for which growth methods exist.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez & Eelke Jongejans

Examples

```
dff <- generateData()
gr1 <- makegrowthObjHossfeld(dff)
```

`makeIntegerFmatrix` *Builds P and F matrices built off regressions fitted to discrete variables (with probability mass functions rather than probability density functions).*

Description

Uses survival, growth and fecundity objects to construct a matrix defining per-capita contribution to recruitment stages (e.g., propagules [seed, spore], seedlings, calves) by reproductive stages due to sexual reproduction (for the F matrix); and transition probabilities determined from survival and growth to quasi-continuous stages based on integer data (e.g. number of leaves) as opposed to truly continuous data (e.g. mass). NOTE - old createIntegerPmatrix is being deprecated; use makeIntegerPmatrix instead; etc

Usage

```
makeIntegerFmatrix(fecObj, nEnvClass = 1, meshpoints = 1:20,
chosenCov = data.frame(covariate=1),
preCensus = TRUE, survObj = NULL, growObj = NULL, offspringObj=NULL)

makeIntegerPmatrix(nEnvClass = 1,
meshpoints = 1:20,
chosenCov = data.frame(covariate = 1),
growObj, survObj,
discreteTrans = 1)
```

Arguments

`fecObj`	fecundity object.
`nEnvClass`	numeric, number of environmental classes, always = 1 for non-compound matrices.
`meshpoints`	numeric, identifying meshpoints
`chosenCov`	data-frame indicating level of the discrete covariate, or range of values where multiple covariates are modeled.
`preCensus`	logical (TRUE or FALSE), indicating whether the fecundity object should represent an interval between pre-breeding or a post-breeding censuses. defaults to TRUE (pre-breeding census), meaning that all reproduction and offspring rates required for the F matrix are embedded in fecObj. Alternatively, an F matrix

	based on post-breeding census (preCensus=FALSE) requires additional survObj and growObj, to cover the survival and growth of the parents until the reproductive event.
`survObj`	survival object, describing the survival of parents from a census until the reproductive event starts (at some point during the inter-census time step). If preCensus = FALSE but no survival object is provided, it is assumed that all individuals survive to the breeding event.
`growObj`	growth object, describing the growth of parents that survive until the reproductive event starts. Warning: this growth object is still ignored in makeIPMFmatrix in the current version of IPMpack. It will become functional in coming versions. So far it is assumed that at time of breeding the individuals have the same size as at the beginning of the time interval.
`discreteTrans`	object of class discreteTrans, or numeric.
`offspringObj`	growth object, describing the size of offspring (this process may alternatively appear in fecObj).

Details

do check whether the Pmatrix adequately reflects survival by using diagnosticsPmatrix().

Value

an object of class IPMmatrix with dimensions length(meshpoints)*length(meshpoints), or length(meshpoints)+nrow(discrete

Note

With thanks to Dr Alden Griffith.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

For information on F and P matrix: Caswell. 2001. Matrix population models: construction, analysis, and interpretation. 2nd ed. Sinauer. p110-112.

See Also

`makeDiscreteTransInteger`, `makeFecObjInteger`

Examples

```
# Open dataset for the herbaceous perennial Cryptantha flava
# where the state variable is integer (number of rosettes)
data(dataIPMpackCryptantha)
head(dataIPMpackCryptantha)
```

```
d <- dataIPMpackCryptantha

#See the description of the data for information on the variables
help(dataIPMpackCryptantha)

# For this example, focus only on the first annual transition available
# in the dataset
d1 <- d[d$year==2004, ]

#Make survival, growth and fecundity objects assuming a poisson distribution
so <- makeSurvObj(d1)
go1 <- makeGrowthObj(d1, Formula = sizeNext~size, Family = "poisson")
fo <- makeFecObjInteger(d1, Formula = fec1~size, distOffspring = "poisson")

#Create P and F matrices
Pmatrix1 <- makeIntegerPmatrix(growObj = go1, survObj = so, meshpoints = 1:101,
    discreteTrans = 1)
Fmatrix <- makeIntegerFmatrix(fecObj = fo, meshpoints = 1:101)

par(mfrow = c(1, 3), bty = "l")

plot(d1$size, d1$sizeNext, xlab = "Stage at t", ylab = "Stage at t+1")

image(Pmatrix1@meshpoints, Pmatrix1@meshpoints, t(Pmatrix1),
xlab = "Stage at t",
ylab = "Stage at t+1")
image(Fmatrix@meshpoints, Fmatrix@meshpoints, t(Fmatrix),
xlab = "Stage at t",
ylab = "Stage at t+1")

#Same approach, but with negative binomial instead of
# poisson for stage transitions
go2 <- makeGrowthObj(d1, Formula = sizeNext~size, Family = "negbin")

#Recalculate the P matrix
Pmatrix2 <- makeIntegerPmatrix(growObj = go1, survObj = so,
    meshpoints = 1:101, discreteTrans = 1)

par(mfrow = c(1, 3), bty = "l")
plot(d1$size, d1$sizeNext, xlab = "Stage at t", ylab = "Stage at t+1")
points(1:100, predict(go2@fit[[1]], newdata = data.frame(size = 1:100),
type = "response"), type = "l", col = 2)

image(Pmatrix2@meshpoints, Pmatrix2@meshpoints, t(Pmatrix2),
xlab = "Stage at t",
ylab = "Stage at t+1")
image(Fmatrix@meshpoints, Fmatrix@meshpoints, t(Fmatrix),
xlab = "Stage at t",
ylab = "Stage at t+1")

#The following repeats the same approach, but with negative binomial
# instead of poisson for stage transitions
dff <- generateData()
```

```
go2 <- makeGrowthObj(d1, Family = "negbin")
Pmatrix2 <- makeIntegerPmatrix(growObj = go2, survObj = so,
    meshpoints = 1:101, discreteTrans = 1)
```

---

makeIPMCmatrix *Builds C matrices.*

Description

Uses clonality objects to construct a matrix defining per-capita contribution to clonal stages (e.g., propagules [seed, spore], seedlings, calves) by clonal reproduction. NOTE - old createIPMCmatrix is being deprecated; use makeIPMCmatrix instead.

Usage

```
makeIPMCmatrix(clonalObj, nEnvClass = 1, nBigMatrix = 50, minSize = -1,
maxSize = 50, chosenCov = data.frame(covariate=1), integrateType="midpoint",
correction="none",preCensus = TRUE, survObj = NULL, growObj = NULL,
offspringObj=NULL)
```

Arguments

`clonalObj`	clonal reproduction object; currently essentially identical to a fecundity reproduction object
`nEnvClass`	numeric, number of environmental classes, always = 1 for non-compound matrices.
`nBigMatrix`	numeric, number of size bins in the P matrix, defaults to 50.
`minSize`	numeric, minimum size of the P matrix, defaults to -1.
`maxSize`	numeric, maximum size of the P matrix, defaults to 50.
`chosenCov`	data-frame indicating level of the discrete covariate, or range of values where multiple covariates are modeled.
`integrateType`	integration type, defaults to "midpoint" (which uses probability density function); other option is "cumul" (which uses the cumulative density function)
`correction`	correction type, defaults to none. The first option is `constant` which will multiply every column of the IPM by a constant sufficient to adjust values to those predicted for total fertility at that size. The second option is `discretizeExtremes` which will place all transitions to sizes smaller than `minSize` into the smallest bin, and transitions to sizes larger than `maxSize` into the largest bin.
`preCensus`	logical (TRUE or FALSE), indicating whether the fecundity object should represent an interval between pre-breeding or a post-breeding censusses. defaults to TRUE (pre-breeding census), meaning that all reproduction and offspring rates required for the F matrix are embedded in fecObj. Alternatively, an F matrix based on post-breeding census (preCensus=FALSE) requires additional survObj and growObj, to cover the survival and growth of the parents until the reproduction event.

`survObj`	survival object, describing the survival of parents from a census until the reproduction event starts (at some point during the inter-census time step). If preCensus = FALSE but no survival object is provided, it is assumed that all individuals survive to the breeding event.
`growObj`	growth object, describing the growth of parents that survive until the reproduction event starts. Warning: this growth object is still ignored in makeIPMFmatrix in the current version of IPMpack. It will become functional in coming versions. So far it is assumed that at time of breeding the individuals have the same size as at the beginning of the time interval.
`offspringObj`	growth object, describing the size of offspring (this process may alternatively appear in fecObj).

Value

an object of class IPMmatrix of dimensions nBigMatrix or nBigMatrix+nDiscrete classes (defined by clonalObj@offspringSplitter-1).

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

For information on C matrix: Caswell. 2001. Matrix population models: construction, analysis, and interpretation. 2nd ed. Sinauer. p110-112.

For midpoint: Zuidema, Jongejans, Chien, During & Schieving. Integral projection models for trees: a new parameterization method and a validation of model output. Journal of Ecology 98, p345-355.

For multiple-vital rate integration on fecundity: Yang, Jongejans, Yang & Bishop. 2011. The effect of consumers and mutualists of Vaccinum membranaceum at Mount St. Helens: dependence on successional context. PLoS One 10, p1-11.

See Also

`makeIPMPmatrix`,`makeIPMFmatrix`,`makeIPMmatrix`

Examples

```
# Data with only continuous stage and one habitat
dff <- generateData()
dff$fec[dff$fec==0] <- NA
cv1 <- makeClonalObj(dff, Formula = fec~size, Transform = "log")
Cmatrix <- makeIPMCmatrix(clonalObj = cv1, nBigMatrix = 20,
minSize = min(dff$size, na.rm = TRUE), maxSize = max(dff$size, na.rm = TRUE))

image(Cmatrix@meshpoints, Cmatrix@meshpoints, t(Cmatrix),
xlab = "Continuous (e.g. size) stage at t",
ylab = "Continous (e.g. size) stage at t+1")
```

makeIPMFmatrix *Builds F matrices.*

Description

Uses fecundity objects to construct a matrix defining per-capita contribution to recruitment stages (e.g., propagules [seed, spore], seedlings, calves) by reproductive stages due to sexual reproduction. NOTE - old createIPMFmatrix is being deprecated; use makeIPMFmatrix instead.

Usage

```
makeIPMFmatrix(fecObj, nEnvClass = 1, nBigMatrix = 50, minSize = -1,
maxSize = 50, chosenCov = data.frame(covariate=1),
integrateType="midpoint", correction="none",
preCensus = TRUE, survObj = NULL, growObj = NULL, offspringObj=NULL)
```

Arguments

`fecObj`	fecundity object.
`nEnvClass`	numeric, number of environmental classes, always = 1 for non-compound matrices.
`nBigMatrix`	numeric, number of size bins in the F matrix, defaults to 50.
`minSize`	numeric, minimum size of the F matrix, defaults to -1.
`maxSize`	numeric, maximum size of the F matrix, defaults to 50.
`chosenCov`	data-frame indicating level of the discrete covariate, or range of values where multiple covariates are modeled.
`integrateType`	integration type, defaults to "midpoint" (which uses probability density function); other option is "cumul" (which uses the cumulative density function)
`correction`	correction type, defaults to `none`. The first option is `constant` which will multiply every column of the IPM by a constant sufficient to adjust values to those predicted for total fertility at that size. The second option is `discretizeExtremes` which will place all transitions to sizes smaller than `minSize` into the smallest bin, and transitions to sizes larger than `maxSize` into the largest bin.
`preCensus`	logical (TRUE or FALSE), indicating whether the fecundity object should represent an interval between pre-breeding or a post-breeding censusses. defaults to TRUE (pre-breeding census), meaning that all reproduction and offspring rates required for the F matrix are embedded in fecObj. Alternatively, an F matrix based on post-breeding census (preCensus=FALSE) requires additional survObj and growObj, to cover the survival and growth of the parents until the reproduction event.
`survObj`	survival object, describing the survival of parents from a census until the reproduction event starts (at some point during the inter-census time step). If preCensus = FALSE but no survival object is provided, it is assumed that all individuals survive to the breeding event.

`growObj`	growth object, describing the growth of parents that survive until the reproduction event starts. Warning: this growth object is still ignored in makeIPMFmatrix in the current version of IPMpack. It will become functional in coming versions. So far it is assumed that at time of breeding the individuals have the same size as at the beginning of the time interval.
`offspringObj`	growth object, describing the size of offspring (this process may alternatively appear in fecObj).

Value

an object of class IPMmatrix of dimensions nBigMatrix or nBigMatrix+nDiscrete classes (defined by fecObj@offspringSplitter-1).

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

For information on F matrix: Caswell. 2001. Matrix population models: construction, analysis, and interpretation. 2nd ed. Sinauer. p110-112.

For midpoint: Zuidema, Jongejans, Chien, During & Schieving. Integral projection models for trees: a new parameterization method and a validation of model output. Journal of Ecology 98, p345-355.

For multiple-vital rate integration on fecundity: Yang, Jongejans, Yang & Bishop. 2011. The effect of consumers and mutualists of Vaccinum membranaceum at Mount St. Helens: dependence on successional context. PLoS One 10, p1-11.

For information on unintentional eviction from IPMs (which the various corrections try and account for) see Williams et al. 2012 Avoiding unintentional eviction from integral projection models. Ecology.

See Also

`makeIPMPmatrix`,`makeIPMCmatrix`,`makeIPMmatrix`

Examples

```
# Data with only continuous stage and one habitat
dff <- generateData()
dff$fec[dff$fec==0] <- NA
fv1 <- makeFecObj(dff, Formula = fec~size, Transform = "log")
Fmatrix <- makeIPMFmatrix(fecObj = fv1, nBigMatrix = 20,
minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE), correction="constant")

slotNames(Fmatrix)

image(Fmatrix@meshpoints, Fmatrix@meshpoints, t(Fmatrix),
xlab = "Continuous (e.g. size) stage at t",
ylab = "Continous (e.g. size) stage at t+1")
```

---

makeIPMmatrix *Builds IPM matrices.*

Description

Uses survival/growth, fecundity and optionally clonal kernels to make an IPM kernel.

Usage

```
makeIPMmatrix(Pmatrix,Fmatrix,Cmatrix=NULL)
```

Arguments

`Pmatrix`	A survival/growth kernel constructed with makeIPMPmatrix().
`Fmatrix`	A fecundity kernel constructed with makeIPMFmatrix().
`Cmatrix`	A clonal kernel, constructed with makeIPMCmatrix(). Defaults to NULL since clonal reproduction may not be applicable for many species.

Details

A convenience function that makes an IPM kernel from the component kernels with all the same slots. All kernels being combined must have the same dimension, i.e. dim(Pmatrix@.Data) is the same as dim(Fmatrix@.Data).

Value

an object of class IPMmatrix with dimensions nBigMatrix*nBigMatrix, or nbig.matrix+nrow(discreteTrans).

Author(s)

Cory Merow, C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Easterling, Ellner & Dixon. 2000. Size-specific sensitivity: a new structured population model. Ecology 81, p694-708.

Ellner & Rees. 2006. Integral projection models for species with complex demography. The American Naturalist 167, p410-428.

For information on P matrix: Caswell. 2001. Matrix population models: construction, analysis, and interpretation. 2nd ed. Sinauer. p110-112.

For information on unintentional eviction from IPMs (which the various corrections try and account for) see Williams et al. 2012 Avoiding unintentional eviction from integral projection models. Ecology.

See Also

makeIPMPmatrix,makeIPMFmatrix,makeIPMCmatrix, diagnosticsPmatrix, makeDiscreteTrans

Examples

```
dff <- generateData()
Pmatrix <- makeIPMPmatrix(
growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff),
nBigMatrix=20,
minSize = min(dff$size, na.rm = TRUE),
maxSize=max(dff$size, na.rm=TRUE))
dff$fec[dff$fec==0] <- NA
fv1 <- makeFecObj(dff, Formula = fec~size, Transform = "log")
Fmatrix <- makeIPMFmatrix(
fecObj = fv1,
nBigMatrix = 20,
minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE))
IPMmatrix <- makeIPMmatrix(Pmatrix,Fmatrix)
slotNames(IPMmatrix)

require(fields)
par(mfrow=c(2,2))
image.plot(IPMmatrix@meshpoints, IPMmatrix@meshpoints, t(Pmatrix),
xlab = "Size(t)",
ylab = "Size(t+1)",
main = "Survival/Growth Kernel")
image.plot(IPMmatrix@meshpoints, IPMmatrix@meshpoints, t(Fmatrix),
xlab = "Size (t)",
ylab = "Size(t+1)",
main = "Fecundity Kernel")
image.plot(IPMmatrix@meshpoints, IPMmatrix@meshpoints, t(IPMmatrix),
xlab = "Size(t)",
ylab = "Size(t+1)",
main = "IPM Kernel")
# trick to visualize the whole IPM kernel when the Fmatrix has values>>Pmatrix
image.plot(IPMmatrix@meshpoints, IPMmatrix@meshpoints, t(IPMmatrix)^.1,
xlab = "Size(t)",
ylab = "Size(t+1)",
main = "IPM Kernel^(.01)")
```

---

makeIPMPmatrix *Builds P matrices.*

Description

Uses growth and survival objects to construct a matrix defining probabilities for transitions between sizes due to growth and survival. Extensions for transition to discrete classes are possible. NOTE - old createIPMPmatrix is being deprecated; use makeIPMPmatrix instead.

Usage

```
makeIPMPmatrix(nEnvClass = 1, nBigMatrix = 50,
minSize = -1, maxSize = 50, chosenCov = data.frame(covariate=1),
growObj, survObj, discreteTrans=1,
integrateType = "midpoint", correction="none")
```

Arguments

`nEnvClass`	numeric, number of environmental classes, always = 1 for non-compound matrices.
`nBigMatrix`	numeric, number of size bins in the P matrix, defaults to 50.
`minSize`	numeric, minimum size of the P matrix, defaults to -1.
`maxSize`	numeric, maximum size of the P matrix, defaults to 50.
`chosenCov`	data-frame indicating level of the discrete covariate, or range of values where multiple covariates are modeled.
`growObj`	growth object.
`survObj`	survival object.
`discreteTrans`	object of class discreteTrans, or numeric.
`integrateType`	integration type, defaults to "midpoint" (which uses probability density function); other option is "cumul" (which uses the cumulative density function).
`correction`	correction type, defaults to `none`. The first option is `constant` which will multiply every column of the IPM by a constant sufficient to adjust values to those predicted for survival at that size. The second option is `discretizeExtremes` which will place all transitions to sizes smaller than `minSize` into the smallest bin, and transitions to sizes larger than `maxSize` into the largest bin.

Details

The number of bins (nBigMatrix) is combined with the minimum and maximum size to define the meshpoints of the IPM. Bins should be sufficient and the size range should encompass the size range of the data. If a "discreteTrans" exists, then discrete stages will be added to the IPM structure. If multiple discrete covariate levels are available, chosenCov identifies the covariate value for which an IPM is required; if a series of covariates are being modeled, chosenCov is a vector of these covariates, and growth will reflect these values.

Value

an object of class IPMmatrix with dimensions nBigMatrix*nBigMatrix, or nbig.matrix+nrow(discreteTrans).

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Easterling, Ellner & Dixon. 2000. Size-specific sensitivity: a new structured population model. Ecology 81, p694-708.

Ellner & Rees. 2006. Integral projection models for species with complex demography. The American Naturalist 167, p410-428.

For information on P matrix: Caswell. 2001. Matrix population models: construction, analysis, and interpretation. 2nd ed. Sinauer. p110-112.

For information on unintentional eviction from IPMs (which the various corrections try and account for) see Williams et al. 2012 Avoiding unintentional eviction from integral projection models. Ecology.

See Also

makeIPMFmatrix,makeIPMmatrix, diagnosticsPmatrix, makeDiscreteTrans

Examples

```
dff <- generateData()
Pmatrix <- makeIPMPmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize=max(dff$size, na.rm=TRUE), growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff))

image(Pmatrix@meshpoints, Pmatrix@meshpoints, t(Pmatrix),
xlab = "Continuous (e.g. size) stage at t",
ylab = "Continuous (e.g. size) stage at t+1")

Pmatrix <- makeIPMPmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize=max(dff$size, na.rm=TRUE),
   growObj = makeGrowthObj(dff,regType="changingVar"),
survObj = makeSurvObj(dff))

image(Pmatrix@meshpoints, Pmatrix@meshpoints, t(Pmatrix),
xlab = "Continuous (e.g. size) stage at t",
ylab = "Continuous (e.g. size) stage at t+1")

#example with discrete transition matrix
dff <- generateData(type="discrete")
Pmatrix <- makeIPMPmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize=max(dff$size, na.rm=TRUE), discreteTrans=makeDiscreteTrans(dff),
growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff))

#discrete stages not plotted
image(Pmatrix@meshpoints, Pmatrix@meshpoints,
t(log(Pmatrix[2:length(Pmatrix@meshpoints),2:length(Pmatrix@meshpoints)])),
xlab = "Continuous (e.g. size) stage at t",
ylab = "Continuous (e.g. size) stage at t+1")
```

`makeOffspringObj` *Function to build offspring objects*

Description

A function that fits regressions that define offspring size and from these build growth objects for which methods to build an IPM object are defined; alternatively, the function creates offspring objects following a specified formula with specified coefficients and sd. Note that if an "offspringNext" column is available in the data, the data will be subsetted based on whether offspringType="sexual" or "clonal" (the default is "sexual"); - otherwise, appropriate data must be supplied.

Usage

```
makeOffspringObj(dataf=NULL, Formula = sizeNext ~ size,
regType = "constantVar", Family = "gaussian", link = NULL,
coeff=NULL,sd=NULL, reproductionType="sexual")
```

Arguments

`dataf`	a dataframe with columns 'size' and 'sizeNext'('size' is size of parent at t, and may not be known, 'sizeNext' is offspring size at t+1); facultatively, dataf may include 'covariate' and 'covariatel' for a single discrete covariate, indicating values at t, and at t+1, respectively; these must take values of sequential integers, starting at '1'.
`Formula`	a formula describing the desired explanatory variables (interactions, etc) according to the R notation for `formula`. style, i.e. separated by '+', '*', ':' and response variable. Possible covariates include 'size', 'size2' (size^2), 'size3' (size^3),'logsize' (log(size)), 'logsize2' (log(size)^2), and 'covariate' (if this name is used, the assumption is made that this is a discrete covariate from which compound matrices may be constructed); or any other covariate available in dataf.
`regType`	possible values include 'constantVar' or 'changingVar'
`Family`	possible values include 'gaussian', 'poisson', 'negbin'
`link`	defaults to NULL, currently only relevant with Family="negbin", and only "log" and "identity" are permitted
`coeff`	numeric vector of required coefficients to be imposed if dataf is NULL; must be compatible with Formula
`sd`	numeric of required sd to be imposed if dataf is NULL
`reproductionType`	whether the relationship should be fitted for sexual or clonal offspring; the default is "sexual"; this will only be relevant if data is provided and has a column "offspringNext"

Value

An object of class growthObj, or growthObjPois, growthObjIncr, or growthObjLogIncr; or growthObjDeclineVar, or growthObjIncrDeclineVar, or growthObjLogIncrDeclineVar. These are S4 objects which contain the slots:

`fit`	an object of class `lm` or `glm` or `gls` that can be used with predict in the growth methods

Slots can be listed by using slotNames(growthObj)

Note

Note that DeclineVar objects cannot currently be constructed without a data-frame.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez & Eelke Jongejans

See Also

`makeGrowthObj`, `makeSurvObj`, `makeFecObj`

Examples

```
#generate data
dff <- generateData()

#make simple linear regression growth object relating size to size at t+1
off1 <- makeOffspringObj(dataf = dff, Formula = sizeNext~1)

#now specifying parameters and supplying no data
off1 <- makeOffspringObj(Formula = incr ~ size + covariate,coeff=c(1,1,1),sd=1)

##make an Fmatrix with this
dff<-generateData()
dff$fec[dff$fec==0] <- NA

off1 <- makeOffspringObj(dataf = dff, Formula = sizeNext~1)
fv1 <- makeFecObj(dff, Formula = fec~size, Transform = "log",
    doOffspring=FALSE)
Fmatrix1 <- makeIPMFmatrix(fecObj = fv1, nBigMatrix = 20,
minSize = min(dff$size, na.rm = TRUE), maxSize = max(dff$size,
na.rm = TRUE), correction="constant", offspringObj=off1)

#compare with the other approach (where offspring object is not separate)
fv2 <- makeFecObj(dff, Formula = fec~size, Transform = "log",
offspringSizeExplanatoryVariables = "1", doOffspring=TRUE)
Fmatrix2 <- makeIPMFmatrix(fecObj = fv2, nBigMatrix = 20,
minSize = min(dff$size, na.rm = TRUE), maxSize = max(dff$size,
na.rm = TRUE), correction="constant", offspringObj=NULL)

par(mfrow=c(1,2))
```

```
image(Fmatrix1); image(Fmatrix2)
```

`makeSurvObj` *Functions to build survival objects*

Description

A function to fit logistic regressions defining survival following user defined formulas (e.g., size+size^2, etc) to build survival objects for which methods to build an IPM object are defined.

Usage

```
makeSurvObj(dataf,Formula=surv~size+size2, coeff=NULL)
```

Arguments

`dataf`	a dataframe with columns 'size' and 'surv'('size' is size at t, 'surv' is 0 for death of the individual and 1 for survival); facultatively, dataf may include 'covariate' and 'covariatel' for a single discrete covariate, indicating values at t, and at t+1, respectively; these must take values of sequential integers, starting at '1'.
`Formula`	a formula describing the desired explanatory variables (interactions, etc) in classical R style, i.e. separated by '+', '*', ':'. Possible covariates include 'size', 'size2' (size^2), 'size3' (size^3),'logsize' (log(size)), 'logsize2' (log(size)^2), and 'covariate'. Response should be 'surv' to match `dataf`
`coeff`	numeric vector of required coefficients to be imposed if dataf is NULL; must be compatible with Formula

Value

An object of class survObj which is a S4 object which contains the slots:

`fit`	an object of class `lm` or `glm` that can be used with predict in the survival methods

Slots can be listed by using slotNames(survObj)

Note

See manual for details on building case-specific survival objects.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez & Eelke Jongejans

See Also

picSurv

Examples

```
#generate data
dff <- generateData()
#make simple logistic regression survival object relating survival to size at t
sv1 <- makeSurvObj(dff, Formula=surv~size)
#assess fit for model with discrete environmental classes fitted
sv1 <- makeSurvObj(dff, Formula=surv~size+covariate)

#now specifying parameters and supplying no data
sv1 <- makeSurvObj(Formula = surv ~ size + covariate,coeff=c(1,1,1))
```

`meanLifeExpect` *Calculates the mean life expectancy.*

Description

Provided a P matrix, which defines survival transitions across stages, this function outputs a vector defining life expectancy in units of the time step used (see convertIncrement()), for each of the size bins.

Usage

```
meanLifeExpect(IPMmatrix)
```

Arguments

`IPMmatrix`	an IPMmatrix object defining survival transitions.

Details

Note that more complex approaches for discretely varying environments (e.g., as in Tuljapurkar & Horvitz 2006.) have yet to be implemented.

Value

A vector of life expectancies each corresponding to a value of the size bins defined by Pmatrix@meshpoints.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Caswell, 2001. Matrix population models: analysis, construction and interpretation. 2nd ed. Sinauer. p118-120.

Cochran & Ellner. 1992. Simple methods for calculating age-based life history parameters for stage-structured populations. Ecological Monographs 62, p345-364.

Tuljapurkar & Horvitz. 2006. From stage to age in variable environments. Life expectancy and survivorship. Ecology 87, p1497-1509.

See Also

makeIPMPmatrix

Examples

```
# With a single continuous state variable (e.g. size):
dff <- generateData()
Pmatrix <- makeIPMPmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE), growObj=makeGrowthObj(dff),
survObj = makeSurvObj(dff))
meanLifeExpect(Pmatrix)

Pmatrix <- makeIPMPmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE), growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff))

plot(meanLifeExpect(Pmatrix), ylab = "Mean life expectancy",
xlab = "Continuous (e.g. size) stage", type = "l", col="dark gray",
ylim = c(0,max(meanLifeExpect(Pmatrix))))

# With continuous (e.g. size) and discrete (e.g. seedbank) stages:
dff <- generateData(type="discrete")
dff$covariate <- sample(1:3, size = nrow(dff), replace = TRUE)
dff$covariateNext <- sample(1:3, size = nrow(dff), replace = TRUE)
discM <- makeDiscreteTrans(dff)
Pmatrix <- makeCompoundPmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE), envMatrix = makeEnvObj(dff),
growObj = makeGrowthObj(dff, Formula = sizeNext~size+size2+covariate),
survObj = makeSurvObj(dff, Formula = surv~size+size2+covariate),
discreteTrans = discM)
mLE <- meanLifeExpect(Pmatrix)

# showing three environments on different panels,
# life expectancy of discrete stages
# shown at level of the first size class
par(mfrow=c(max(Pmatrix@env.index),1))

xvals <- c(rep(Pmatrix@meshpoints[1],ncol(discM@discreteTrans)-1),
    Pmatrix@meshpoints)

for (k in 1:max(Pmatrix@env.index)) {
indx <- ((k-1)*(ncol(discM@discreteTrans)-1+length(Pmatrix@meshpoints))+1):
```

```
(k*(ncol(discM@discreteTrans)-1+length(Pmatrix@meshpoints)))

plot(xvals,mLE[indx],
ylab = "Mean life expectancy",
xlab = "Continuous (e.g. size) and discrete (e.g. seedbank) stages",
type = "l", col = "dark gray", ylim = c(0,max(mLE)),
main=paste("habitat ",k,sep=""))
}
```

`passageTime` *Defines passage time to a chosen continuous stage.*

Description

Estimates the time in units of the chosen time-steps (see convertIncrement()) that it will take to reach a chosen continuous (e.g. size) stage for the first time conditional on surviving from each of the meshpoints of the IPM; currently not defined for matrices with discrete as well as continuous stage categories.

Usage

```
passageTime(chosenSize, IPMmatrix)
```

Arguments

`chosenSize`	numeric, the target size.
`IPMmatrix`	an IPMmatrix object describing growth-survival transitions (a P matrix).

Details

Passage time for values exactly equal to the chosen size (targetSize) are one year, because of way the conditionals are framed. Values slightly less than the target size may on average take longer due to variance in growth, mortality, leading to discontinuities in the pattern of passage time over age. Passage time from values > than targetSize should be ignored (space to the right of the red vertical line in example below), unless dealing with an organism that is able to display retrogression. Use stochPassageTime for compound matrices.

Value

A vector of times in the units of the chosen time-steps corresponding to each of the IPM meshpoints.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Caswell, 2001. Matrix population models: analysis, construction and interpretation. 2nd ed. Sinauer. p119.

Metcalf, Horvitz, Tuljapurkar & Clark. 2009. A time to grow and a time to die: a new way to analyze the dynamics of size, light, age and death of tropical trees. Ecology 90, p2766-2778.

For bias in this estimation where variance in growth is small relative to the size range: Zuidema, Jongejans, Chien, During & Schieving. 2010. Integral Projection Models for trees: a new parameterization method and a validation of model output. Journal of Ecology 98, p345-355.

For species with shrinkage: Salguero-Gomez & Casper. 2010. Keeping shrinkage in the demographic loop. Journal of Ecology 98, p313-323.

See Also

`meanLifeExpect`, ~~~

Examples

```
# With a single continuous state variable (e.g. size)
dff <- generateData()
Pmatrix <- makeIPMPmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE), growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff), correction="constant")
targetSize <- 8
passage <- passageTime(targetSize, Pmatrix)

plot(Pmatrix@meshpoints, passage, ylab = "Passage time",
 xlab = "Continuous (e.g. size) stage",
type = "l", col = "dark gray", ylim = c(0, max(passage)),
xlim=c(Pmatrix@meshpoints[1],targetSize+1))
abline(v = targetSize, col="red")
```

`picGrow` *Makes pictures of data with growth models*

Description

Takes the data file and a growth object and shows the model fit over the data.

Usage

```
picGrow(dataf, growObj, mainTitle = "Growth",...)
```

Arguments

`dataf`	a dataframe with columns 'size' and 'sizeNext'('size' is continuous stage variable at t, 'sizeNext' is stage variable at t+1); facultatively, dataf may include 'covariate' and 'covariateNext' for a single discrete covariate, indicating values at t, and at t+1, respectively; these must take values of sequential integers, starting at '1'. For models fitting growth increment, 'incr' or 'logincr' may be directly provided as a column in the dataframe, otherwise they are calculated as dataf$sizeNext-dataf$size or log(dataf$sizeNext - dataf$size), respectively.
`growObj`	an object of class growthObj that contains a fit for which R has methods for the function "predict".
`mainTitle`	a character string that will be used as in the 'main' argument of plot. Defaults to 'Growth'.
`...`	other arguments to `plot`

Details

Note that this model will only work with growth objects that contain objects of class glm or lm, i.e. ones for which R has defined methods for the function "predict".

Value

Returns nothing.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

See Also

`makeGrowthObj`, `picSurv`

Examples

```
dff <- generateData()
gr1 <- makeGrowthObj(dff)
picGrow(dff, gr1)
```

`picSurv` *Makes pictures of survival.*

Description

Produces figures of value for assessing survival fit given data.

Usage

```
picSurv(dataf, survObj, ncuts = 20, makeTitle = "Survival", ...)
```

Arguments

`dataf`	a dataframe with columns 'size' and 'sizeNext'('size' is continuous stage variable at t, 'sizeNext' is continuous stage at t+1); facultatively, dataf may include 'covariate' and 'covariateNext' for a single discrete covariate, indicating values at t, and at t+1, respectively; these must take values of sequential integers, starting at '1'. For models fitting growth increment, 'incr' or 'logincr' may be directly provided as a column in the dataframe, otherwise they are calculated as dataf$sizeNext-dataf$size or log(dataf$sizeNext-dataf$size), respectively.
`survObj`	an object of class survObj.
`ncuts`	number of consecutive values for which means of survival and continuous (e.g. size) stage are taken for the plotting.
`makeTitle`	character that defines title, defaults to "Survival"
`...`	extra arguments to plot (e.g, ylim, etc).

Value

Returns nothing.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

See Also

`makeSurvObj`, `picGrow`

Examples

```
dff <- generateData()
sv1 <- makeSurvObj(dff)
picSurv(dff,sv1)
```

`plotGrowthModelComp` *Plots compared models built with* `growthModelComp` *and* `survModelComp`.

Description

Function plots compared models built with `growthModelComp` and `survModelComp`. This can be invoked directly from `growthModelComp` and `survModelComp` with the argument `makePlot = TRUE`.

Usage

```
plotGrowthModelComp(grObj,summaryTable, dataf, expVars,
testType = "AIC",
plotLegend = TRUE, mainTitle = "", legendPos = "topright",...)

plotSurvModelComp(svObj, summaryTable, dataf, expVars,
testType = "AIC",  plotLegend = TRUE, mainTitle = "",ncuts=20,
legendPos = "bottomleft",...)
```

Arguments

`grObj`	a list with the objects of the class growth object equal to `treatN`.
`svObj`	a list with the objects of the class survival object equal to `treatN`.
`summaryTable`	dataframe output from `growthModelComp` and `survModelComp` that contains linear predictor and testType scores (see `growthModelComp` and `survModelComp`).
`dataf`	dataframe containing `size` and `sizeNext`
`expVars`	vector, list of covariates. Defaults to `c("1",  "size", "size + size2")`.
`testType`	character string identifying the metric used to compare models. Can be any string that uses `loglike` from the lm or glm object. For example `"AIC"` or `"BIC"`. Defaults to `"AIC"`.
`plotLegend`	logical indicated whether a legend is created. If `TRUE`, positions the legend in `"topleft"` for growth models and `"bottomleft"` for survival models.
`mainTitle`	string to place as the `main` attribute in plots (if `makePlot = TRUE`. defaults to `NULL`.
`ncuts`	number of consecutive size values for which to take means of size and survival for plotting.
`legendPos`	position of the legend on the figure ("topright", "bottomleft", ...)
`...`	additional arguments to plot (ylim, col, etc)

Details

Plots multiple growth and survival objects returned from `growthModelComp` and `survModelComp`. See `plotGrowthModelComp` and `plotSurvModelComp` for more details.

Value

a plot object.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

See Also

`growthModelComp`,`growthModelComp`

Examples

```
# Data with size and sizeNext
dff <- generateData()

grModels <- growthModelComp(dff, makePlot = FALSE)
```

---

predictFutureDistribution

Predicts continuous (e.g. size) stage distribution in the future giving current population's stage distribution.

Description

Function to project a population forwards using an IPM and a starting environment. The IPM may be structured by continuous (e.g. size) stage alone, or by continuous stage and environment.

Usage

```
predictFutureDistribution(startingSizes, IPM, n.time.steps, startingEnv = 1)
```

Arguments

`startingSizes`	vector containing the sizes of the desired starting population.
`IPM`	an IPMmatrix object (P matrix if only interested in survival projections, P matrix+ F matrix otherwise).
`n.time.steps`	a numeric defining the number of time steps for which projection is required.
`startingEnv`	vector defining the desired starting environment, of length one or length startingSizes; ignored if no environmental states are provided; otherwise if the length is less than startingSizes assumes all individuals start in the same environment, given by startingEnv[1].

Details

Currently this does not accept IPMs with discrete stages (e.g. seedbank).

Value

`n.new.dist0`	starting frequency distribution along meshpoints in IPMmatrix.
`n.new.dist`	final frequency distribution.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

Examples

```
# Define starting population of interest
startPop <- rnorm(1000,2,1)

# Build T and F matrix
dff <- generateData()
Pmatrix <- makeIPMPmatrix(minSize = 1.1*min(dff$size, na.rm = TRUE),
maxSize = 1.1*max(dff$size, na.rm = TRUE), growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff), correction = "constant")
Fmatrix <- makeIPMFmatrix(minSize = 1.1*min(dff$size, na.rm = TRUE),
maxSize = 1.1*max(dff$size, na.rm = TRUE),
fecObj = makeFecObj(dff, fecConstants = data.frame(est=0.7), Transform="log"),
correction="constant")

# Make an IPMmatrix object containing P matrix + F matrix
# by replacing the P matrix
IPM <- Pmatrix
IPM@.Data <- Pmatrix + Fmatrix

# Project population five steps
a5 <- predictFutureDistribution(startingSizes = startPop, IPM = IPM,
n.time.steps = 5, startingEnv = 1)
```

`R0Calc` *Calculates net reproductive rate (R0) from an IPM.*

Description

Estimates lifetime reproductive success from a full IPM, including survival, growth and fecundity.

Usage

```
R0Calc(Pmatrix, Fmatrix)
```

Arguments

`Pmatrix`	a matrix (not necessarily of class IPMmatrix).
`Fmatrix`	a matrix (not necessarily of class IPMmatrix).

Value

numeric

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Caswell, 2001. Matrix population models: analysis, construction and interpretation. 2nd ed. Sinauer. p126.

See Also

makeIPMPmatrix,makeIPMFmatrix

Examples

```
dff <- generateData()
Pmatrix <- makeIPMPmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE), growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff), correction="constant")
Fmatrix <- makeIPMFmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE),
fecObj = makeFecObj(dff,Formula=fec~size), correction="constant")
R0Calc(Pmatrix, Fmatrix)
```

---

sampleIPM	*Builds list of IPMs or P matrices from list growth, survival, fecundity and discreteTrans objects. It is helpful when building multiple IPMs for study of parameter uncertainty or stochastic dynamics.*

Description

Uses lists of vital rate objects to create a list of IPM or P matrices.

Usage

```
sampleIPM( growObjList=NULL,survObjList=NULL,fecObjList=NULL,
    offspringObjList=NULL, discreteTransList=1,
    nBigMatrix,minSize,maxSize,
    covariates=FALSE,envMat=NULL,
    integrateType="midpoint",correction="none",warn=TRUE)
```

Arguments

growObjList	list of growth objects.
survObjList	list of survival objects.
fecObjList	list of fecundity objects.
offspringObjList	list of survival objects.

`discreteTransList`	list of survival objects.
`nBigMatrix`	number of meshpoints.
`minSize`	minimum size.
`maxSize`	maximum size.
`covariates`	level of the covariate.
`envMat`	environmental matrix - defaults to NULL.
`integrateType`	integration type, defaults to "midpoint" (which uses probability density function); other option is "cumul" (which uses the cumulative density function).
`correction`	correction type, defaults to "none"; option is "constant" which will multiply every column of the IPM by a constant sufficient to adjust values to those predicted for survival at that size.
`warn`	turn warning messages on/off.

Author(s)

Cory Merow, C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans.

See Also

`sampleVitalRateObj`,`sampleIPMOutput`,`sampleSequentialIPMs`

Examples

```
# =============================================================================
# Sample Vital Rate Objects
# Parametric bootstrap sample for a growth object
dff <- generateData(type='discrete')
gr1 <- makeGrowthObj(dff)
gr1List=sampleVitalRateObj(gr1,nSamp=9)

# Parametric bootstrap sample for a survival object
sv1 <- makeSurvObj(dff)
sv1List=sampleVitalRateObj(sv1,nSamp=9)

# Parametric bootstrap sample for a fecundity object
fv1 <- makeFecObj(dff)
fv1List=sampleVitalRateObj(
fv1,nSamp=9,
nDiscreteOffspringTransitions =100,
nOffspring=100)

# Parametric bootstrap sample for a discrete transition object
dt1 <- makeDiscreteTrans(dff)
dt1List=sampleVitalRateObj(
dt1,nSamp=9,
nDiscreteGrowthTransitions=100)
# =============================================================================
# Make a list of growth/survival (P) matrices (omitting fecundity)
```

```
Pmatrixlist=sampleIPM(
growObjList=gr1List,
survObjList=sv1List,
fecObjList =NULL,
nBigMatrix = 20, minSize = -5, maxSize = 20)
# plot results
par(mfrow=c(3,3))
lapply(Pmatrixlist,image)

# Combine the list of fecundity objects with a single survival
# and growth object in a list of IPMs to look at just the impact
# of uncertainty in fecundity parameter estimates on population growth rate
IPMlist2=sampleIPM(
growObjList=list(gr1),
survObjList=list(sv1),
fecObjList =fv1List,
discreteTransList=list(dt1),
nBigMatrix = 20, minSize = -5, maxSize = 20)
# plot results
lapply(IPMlist2,image)

# Combine the lists of all vital rate objects in a list of IPMs to
# look at the impact of uncertainty in all parameters on
# population growth rate
IPMlist3=sampleIPM(
growObjList=gr1List,
survObjList=sv1List,
fecObjList =fv1List,
discreteTransList=list(dt1),
nBigMatrix = 20, minSize = -5, maxSize = 20)
# plot results
lapply(IPMlist3,image)

# ============================================================================
# Summarize the outputs
# Get uncertainty in passage time from the list of growth/survival matrices
IPMout1=sampleIPMOutput(PMatrixList=Pmatrixlist)
qLE=apply(IPMout1[['LE']],2,quantile,probs=c(.025,.5,.975))
plot(IPMout1$meshpoints,qLE[2,],type='l',ylim=c(0,max(qLE)))
lines(IPMout1$meshpoints,qLE[1,],type='l',lty=3)
lines(IPMout1$meshpoints,qLE[3,],type='l',lty=3)

# Get uncertainty in lambda from the list of IPMs where only fecundity
# varied
IPMout2=sampleIPMOutput(IPMList=IPMlist2)
qlambda=quantile(IPMout2[['lambda']],probs=c(.025,.5,.975))
boxplot(IPMout2[['lambda']])

# Get uncertainty in lambda and passage time from size 5
# to a series of size from the list of IPMs where all vital rates varied
IPMout3=sampleIPMOutput(
IPMList=IPMlist3,
passageTimeTargetSize=c(10),
sizeToAgeStartSize=c(5),
sizeToAgeTargetSize=c(6,7,8,9,10))
qlambda=quantile(IPMout3[['lambda']],probs=c(.025,.5,.975))
boxplot(IPMout3[['resAge']])
```

---

sampleIPMOutput

Gets IPM output from a list of P matrices (only survival and size information) or full IPMs (P matrices + F matrices; the latter include sexual reproduction information).

Description

Gets synthetic values including life expectancy, passage time, and, if a fecundity matrix is available, population growth rate (lambda), stable stage distribution, reproductive output, etc. It is helpful when building multiple IPMs for study of parameter uncertainty or stochastic dynamics.

Usage

```
sampleIPMOutput(IPMList=NULL,PMatrixList=NULL,passageTimeTargetSize=c(),
sizeToAgeStartSize=c(),sizeToAgeTargetSize=c(),warn=TRUE)
```

Arguments

`IPMList`	List of survival-size IPM matrices for which summary statistics desired. When this information is included, population growth rate (lambda), and stable stage distribution will be provided.
`PMatrixList`	List of survival-growth (P) matrices for which summary statistics desired. When this information is included, only passage time and sizeToAge calculations will be provided
`passageTimeTargetSize`	Target size for passage time. If none is provided defaults to the median of the IPM meshpoints.
`sizeToAgeStartSize`	Starting size to determine the expected age which sizeToAgeTargetSize will be reached. If none is provided defaults to the minimum of the IPM meshpoints.
`sizeToAgeTargetSize`	Target size to determine the expected age which sizeToAgeTargetSize will be reached. If none is provided defaults to the IPM meshpoints.
`warn`	turn warning messages on/off.

Value

`LE`	matrix of life expectancies, columns correspond to meshpoints, rows corresponding to each element of the list of P matrices

`pTime`	matrix of passage times to the targetSize from each of the meshpoints (columns) and for each element in the P matrix list (columns).
`lambda`	vector of population growth rates corresponding to value obtained combining each element of the list of P matrices with the corresponding element in the list of F matrices; if no F matrix list is provided, it returns a vector of NAs.
`stableStage`	matrix of stable stage distributions rows corresponding to values obtained. combining each element of the list of Pmatrices with the corresponding element in the list of Fmatrices; if no Fmatrix list is provided, this is a matrix of NAs.
`meshpoints`	matrix meshpoints.
`resSize`	matrix providing target sizes for size to age estimate (assuming age=1 at sizeStart), of length nsizeToAge space equally between the smallest and largest meshpoints.
`resAge`	matrix providing time in time-steps to get to resSize, rows corresponding sequential elements in the list of P matrices.

Note

This function has replaced the functionality of getIPMoutput and getIPMoutputDirect. Those functions are no longer supported but have been hidden (.getIPMoutput and .getIPMoutputDirect) and can be accessed for backward compatibility.

Author(s)

Cory Merow, C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans.

References

Zuidema, Jongejans, Chien, During & Schieving. Integral projection models for trees: a new parameterization method and a validation of model output. Journal of Ecology 98, p345-355.

See Also

`sampleIPM`, `sampleVitalRateObj`, `sampleSequentialIPMs`

Examples

```
# =============================================================================
# Sample Vital Rate Objects
# Parametric bootstrap sample for a growth object
dff <- generateData(type='discrete')
gr1 <- makeGrowthObj(dff)
gr1List=sampleVitalRateObj(gr1,nSamp=9)

# Parametric bootstrap sample for a survival object
sv1 <- makeSurvObj(dff)
sv1List=sampleVitalRateObj(sv1,nSamp=9)

# Parametric bootstrap sample for a fecundity object
fv1 <- makeFecObj(dff)
```

```
fv1List=sampleVitalRateObj(
fv1,nSamp=9,
nDiscreteOffspringTransitions =100,
nOffspring=100)

# Parametric bootstrap sample for a discrete transition object
dt1 <- makeDiscreteTrans(dff)
dt1List=sampleVitalRateObj(
dt1,nSamp=9,
nDiscreteGrowthTransitions=100)
# ============================================================================
# Make a list of growth/survival (P) matrices (omitting fecundity)
Pmatrixlist=sampleIPM(
growObjList=gr1List,
survObjList=sv1List,
fecObjList =NULL,
nBigMatrix = 20, minSize = -5, maxSize = 20)
# plot results
par(mfrow=c(3,3))
lapply(Pmatrixlist,image)

# Combine the list of fecundity objects with a single survival
# and growth object in a list of IPMs to look at just the impact
# of uncertainty in fecundity parameter estimates on population growth rate
IPMlist2=sampleIPM(
growObjList=list(gr1),
survObjList=list(sv1),
fecObjList =fv1List,
discreteTransList=list(dt1),
nBigMatrix = 20, minSize = -5, maxSize = 20)
# plot results
lapply(IPMlist2,image)

# Combine the lists of all vital rate objects in a list of IPMs
# to look at the impact of uncertainty in all parameters on population
# growth rate
IPMlist3=sampleIPM(
growObjList=gr1List,
survObjList=sv1List,
fecObjList =fv1List,
discreteTransList=list(dt1),
nBigMatrix = 20, minSize = -5, maxSize = 20)
# plot results
lapply(IPMlist3,image)

# ============================================================================
# Summarize the outputs
# Get uncertainty in passage time from the list of growth/survival matrices
IPMout1=sampleIPMOutput(PMatrixList=Pmatrixlist)
qLE=apply(IPMout1[['LE']],2,quantile,probs=c(.025,.5,.975))
plot(IPMout1$meshpoints,qLE[2,],type='l',ylim=c(0,max(qLE)))
lines(IPMout1$meshpoints,qLE[1,],type='l',lty=3)
lines(IPMout1$meshpoints,qLE[3,],type='l',lty=3)
```

```
# Get uncertainty in lambda from the list of IPMs where only
# fecundity varied
IPMout2=sampleIPMOutput(IPMList=IPMlist2)
qlambda=quantile(IPMout2[['lambda']],probs=c(.025,.5,.975))
boxplot(IPMout2[['lambda']])

# Get uncertainty in lambda and passage time from size 5 to a
# series of size from the list of IPMs where all vital rates varied
IPMout3=sampleIPMOutput(
IPMList=IPMlist3,
passageTimeTargetSize=c(10),
sizeToAgeStartSize=c(5),
sizeToAgeTargetSize=c(6,7,8,9,10))
qlambda=quantile(IPMout3[['lambda']],probs=c(.025,.5,.975))
boxplot(IPMout3[['resAge']])
```

`sampleSequentialIPMs` *Makes a list of IPMs where there is a discrete covariate.*

Description

Wrapper function to build the IPM corresponding to every level of the discrete covariate, and return a list of these.

Usage

```
sampleSequentialIPMs(dataf, nBigMatrix = 10, minSize = -2,
    maxSize = 10,
integrateType = "midpoint", correction = "none",
explSurv = surv ~ size + size2 + covariate,
explGrow = sizeNext ~ size + size2 + covariate,
regType = "constantVar",
explFec = fec ~size,  Family="gaussian",
Transform = "none",
fecConstants = data.frame(NA))
```

Arguments

`dataf`	a dataframe with columns ‘size’, ‘sizeNext’, ’surv’, ’fec’, ’covariate’, ’covariatel’; and ’age’ indicating which individuals are seedlings for identifying the mean and variance in seedling size.
`nBigMatrix`	number of bins in size.
`minSize`	minimum size.
`maxSize`	maximum size.
`integrateType`	integration type.

`correction`	correction for unintentional eviction (individuals move outside the size range of the IPM). This correction redistributes individuals so that column sums of the IPM match expected survival for that column.
`explSurv`	Formula and explanatory variables used in the survival model.
`explGrow`	explanatory variables used in the growth model.
`regType`	Formula and regression Type for growth (normal density function, truncated, etc).
`explFec`	explanatory variables used in the fecundity.
`Family`	a character vector containing the names of the families to be used for the glms, e.g., binomial, poisson, etc. Again, these must appear *in the order defined by the list of formula*
`Transform`	a character vector containing the names of the transforms to be used for the response variables, e.g., log, sqrt, -1, etc. Again, these must appear *in the order defined by the list of formula*
`fecConstants`	data.frame of constant multipliers for the fecundity model.

Value

list of matrices corresponding to covariates, in order.

Note

Formerly makeListIPMs(). makeListIPMs() is no longer supported but has been hidden (.makeListIPMs()) and can be accessed for backward compatibility.

Author(s)

Cory Merow, C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans.

Examples

```
dff <- generateData()
IPMlist <- sampleSequentialIPMs(dff, Transform="log")
```

`sampleVitalRateObj` *Calculates growth objects reflecting distribution of parameters from lm or glm.*

Description

Generate parametric bootstrap samples for vital rate objects (e.g. class growthObj, survObj, fecObj, etc.) from estimated parameters and the variance covariance matrix that defines them using a multivariate normal distribution. It is helpful when building multiple IPMs for study of parameter uncertainty or stochastic dynamics.

Usage

```
sampleVitalRateObj(
Obj,
nSamp=100,
nDiscreteGrowthTransitions=NULL,
nDiscreteOffspringTransitions = NULL,
nOffspring = NULL)
```

Arguments

`Obj`	a growth or survival object with a slot named "fit" containing an lm or glm, etc., or a fertility object with a slot named "fitFec" for `.getListRegObjectsFec`, likewise.
`nSamp`	desired number of samples from the multivariate normal.
`nDiscreteGrowthTransitions`	number of transitions used to estimate a discreteTrans object. This is used to estimate the correct variance for sampling the discreteTrans object. It is only required if a discreteTransObject is provided.
`nDiscreteOffspringTransitions`	number of transitions used to estimate transition probabilities between discrete offspring stages (stored in the @offspringSplitter slot of a fecObj). This is used to estimate the correct variance for the sampling. It is only required if a fecObject is provided.
`nOffspring`	number of transitions used to the offspring size distribution (stored in the @offspringRel and @offspringsd slots of a fecObj). This is used to estimate the correct variance for the sampling. It is only required if a fecObject is provided.

Value

The output is list of the provided vital rate object with different parameter values in each list element, e.g. a list of growth or survival objects containing an lm or glm; or fertility objects likewise.

Note

This function has replaced the functionality of getListRegObjects and getListRegObjects. Those functions are no longer supported but have been hidden (.getListRegObjects and .getListRegObjects) and can be accessed for backward compatibility.

Author(s)

Cory Merow, C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans.

See Also

`sampleIPM` ,`sampleIPMOutput`,`sampleSequentialIPMs`

Examples

```
# =============================================================================
# Sample Vital Rate Objects
# Parametric bootstrap sample for a growth object
dff <- generateData(type='discrete')
gr1 <- makeGrowthObj(dff)
gr1List=sampleVitalRateObj(gr1,nSamp=9)

# Parametric bootstrap sample for a survival object
sv1 <- makeSurvObj(dff)
sv1List=sampleVitalRateObj(sv1,nSamp=9)

# Parametric bootstrap sample for a fecundity object
fv1 <- makeFecObj(dff)
fv1List=sampleVitalRateObj(
fv1,nSamp=9,
nDiscreteOffspringTransitions =100,
nOffspring=100)

# Parametric bootstrap sample for a discrete transition object
dt1 <- makeDiscreteTrans(dff)
dt1List=sampleVitalRateObj(
dt1,nSamp=9,
nDiscreteGrowthTransitions=100)
# =============================================================================
# Make a list of growth/survival (P) matrices (omitting fecundity)
Pmatrixlist=sampleIPM(
growObjList=gr1List,
survObjList=sv1List,
fecObjList =NULL,
nBigMatrix = 20, minSize = -5, maxSize = 20)
# plot results
par(mfrow=c(3,3))
lapply(Pmatrixlist,image)

# Combine the list of fecundity objects with a single survival
# and growth object in a list of IPMs to look at just the impact
# of uncertainty in fecundity parameter estimates on population
# growth rate
IPMlist2=sampleIPM(
growObjList=list(gr1),
survObjList=list(sv1),
fecObjList =fv1List,
discreteTransList=list(dt1),
nBigMatrix = 20, minSize = -5, maxSize = 20)
# plot results
lapply(IPMlist2,image)

# Combine the lists of all vital rate objects in a list of IPMs
# to look at the impact of uncertainty in all parameters on population
# growth rate
```

```
IPMlist3=sampleIPM(
growObjList=gr1List,
survObjList=sv1List,
fecObjList =fv1List,
discreteTransList=list(dt1),
nBigMatrix = 20, minSize = -5, maxSize = 20)
# plot results
lapply(IPMlist3,image)

# =============================================================================
# Summarize the outputs
# Get uncertainty in passage time from the list of growth/survival matrices
IPMout1=sampleIPMOutput(PMatrixList=Pmatrixlist)
qLE=apply(IPMout1[['LE']],2,quantile,probs=c(.025,.5,.975))
plot(IPMout1$meshpoints,qLE[2,],type='l',ylim=c(0,max(qLE)))
lines(IPMout1$meshpoints,qLE[1,],type='l',lty=3)
lines(IPMout1$meshpoints,qLE[3,],type='l',lty=3)

# Get uncertainty in lambda from the list of IPMs where only
# fecundity varied
IPMout2=sampleIPMOutput(IPMList=IPMlist2)
qlambda=quantile(IPMout2[['lambda']],probs=c(.025,.5,.975))
boxplot(IPMout2[['lambda']])

# Get uncertainty in lambda and passage time from size 5
#to a series of size from the list of IPMs where all vital rates varied
IPMout3=sampleIPMOutput(
IPMList=IPMlist3,
passageTimeTargetSize=c(10),
sizeToAgeStartSize=c(5),
sizeToAgeTargetSize=c(6,7,8,9,10))
qlambda=quantile(IPMout3[['lambda']],probs=c(.025,.5,.975))
boxplot(IPMout3[['resAge']])
```

sensParams — *Estimates sensitivity and elasticity of lambda (or R0, or Life expectancy of a chosen bin) to parameters underlying an IPM.*

Description

Uses perturbation to estimate the sensitivity and elasticity of all the parameters underlying an IPM.

Usage

```
sensParams(growObj, survObj, fecObj=NULL, clonalObj=NULL,
nBigMatrix, minSize, maxSize,
chosenCov = data.frame(covariate = 1), discreteTrans = 1,
integrateType = "midpoint", correction = "none", preCensusFec = TRUE,
postCensusSurvObjFec = NULL, postCensusGrowObjFec = NULL,
preCensusClonal = TRUE, postCensusSurvObjClonal = NULL,
postCensusGrowObjClonal = NULL, delta = 1e-04,
response="lambda", chosenBin=1)
```

Arguments

growObj	a growth object.
survObj	a survival object.
fecObj	a fecundity object (not necessary for life expectancy analysis).
clonalObj	a clonality object (not necessary for life expectancy analysis).
nBigMatrix	numeric, number of bins of size used in the IPM matrix.
minSize	numeric, minimum size used for meshpoints of the IPM matrix.
maxSize	numeric, maximum size used for meshpoints of the IPM matrix.
chosenCov	level or value of the covariate(s) at which sensitivity estimation is desired
discreteTrans	matrix of discrete transitions; or 1 if there is none
integrateType	integration type, defaults to "midpoint" (which uses probability density function); other option is "cumul" (which uses the cumulative density function)
correction	correction type, defaults to none. The first option is constant which will multiply every column of the IPM by a constant sufficient to adjust values to those predicted for total fertility at that size. The second option is discretizeExtremes which will place all transitions to sizes smaller than minSize into the smallest bin, and transitions to sizes larger than maxSize into the largest bin.
preCensusFec	logical (TRUE or FALSE), indicating whether the fecundity object represents an interval between pre-breeding or a post-breeding censusses. Defaults to TRUE (pre-breeding census), meaning that all reproduction and offspring rates required for the F matrix are embedded in fecObj. Alternatively, an F matrix based on post-breeding census (preCensusFec=FALSE) uses postCensusSurvObjFec and postCensusGrowObjFec, to cover the survival and growth of the parents until the reproduction event. (not necessary for life expectancy analysis)
postCensusSurvObjFec	survival object representing the survival of the parents until the reproduction event. If not specified (and preCensusFec = FALSE) it is assumed that all parents survive until the reproduction event. (not necessary for life expectancy analysis)
postCensusGrowObjFec	growth object representing the growth of surviving parents until the reproduction event. If not specified (and preCensusFec = FALSE) it is assumed that the parents do not grow until the reproduction event. (not necessary for life expectancy analysis)
preCensusClonal	logical (TRUE or FALSE), indicating whether the clonality object represents an interval between pre-breeding or a post-'breeding' censusses. Defaults to TRUE (pre-'breeding' census), meaning that all clonal propagation and offspring rates required for the C matrix are embedded in clonalObj. Alternatively, an C matrix based on post-'breeding' census (preCensusClonal=FALSE) uses postCensusSurvObjClonal and postCensusGrowObjClonal, to cover the survival and

	growth of the parents until the clonal propagation event. (not necessary for life expectancy analysis)
`postCensusSurvObjClonal`	survival object representing the survival of the parents until the clonal propagation event. If not specified (and preCensusClonal = FALSE) it is assumed that all parents survive until the clonal propagation event. (not necessary for life expectancy analysis)
`postCensusGrowObjClonal`	growth object representing the growth of surviving parents until the clonal propagation event. If not specified (and preCensusClonal = FALSE) it is assumed that the parents do not grow until the clonal propagation event. (not necessary for life expectancy analysis)
`delta`	size of the perturbation desired
`response`	whether lambda, R0 or life expectancy of a desired bin (lifeExpect with chosenBin) is required
`chosenBin`	for analysis of life expectancy, which bin in the IPM Life expectancy should be compared for

Details

The values returned by sensParam are calculated by first calculating lambda for the chosen IPM; then modifying the focal parameter c by a very small amount, c.new=c*(1+delta) (the default for delta =1e-4, but users may specify the value that they want). The function then rebuilds the T and F matrices, and re-calculates lambda. Sensitivity is calculated as:

sens = df(x)/dx = (lam.new-lam)/(c*delta)

i.e., the function estimates the degree to which a small change in the parameter results in a small change in lambda; and elasticity is calculated as:

elas = sens*c/lam = (lam.new-lam)/(lam*delta)

which corresponds to the proportional change in lambda as an outcome of the proportional change in the parameter; analagous calculations are used for R0 and life expectancy.

NOTE: in previous versions of IPMpack (pre 2.0), the output of this function was mis-aligned.

Value

`sens`	a vector of sensitivities of lambda or other variable with names corresponding to parameters.
`elas`	a vector of elasticities to lambda or other variable with names corresponding to parameters.

Note

Modified following code developed by Rees & Rose 2002 (above).

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Rees and Rose. 2002. Evolution of flowering strategies in Oenothera glazioviana: an integral projection model approach. Proceedings of the Royal Society London Seres B 269, p1509-1515.

See Also

sens, elas

Examples

```
dff <- generateData()

#lambda
res <- sensParams(growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff), fecObj = makeFecObj(dff, Transform="log"),
nBigMatrix = 50, minSize = min(dff$size, na.rm=TRUE),
maxSize = max(dff$size, na.rm = TRUE))

par(mfrow = c(2, 1), bty = "l", pty = "m")
barplot(res$sens,
main = expression("Parameter sensitivity of population growth rate "* lambda),
las = 2, cex.names = 0.5)
barplot(res$elas,
main = expression("Parameter elasticity of population growth rate "* lambda),
las = 2, cex.names = 0.5)

#R0
resR0 <- sensParams(growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff), fecObj = makeFecObj(dff, Transform="log"),
nBigMatrix = 50, minSize = min(dff$size, na.rm=TRUE),
maxSize = max(dff$size, na.rm = TRUE), response="R0")

par(mfrow = c(2, 1), bty = "l", pty = "m")
barplot(resR0$sens,
main = expression("Parameter sensitivity of net reproductive rate R"[0]),
las = 2, cex.names = 0.5)
barplot(resR0$elas,
main = expression("Parameter elasticity of net reproductive rate R"[0]),
las = 2, cex.names = 0.5)

#life expectancy
resLE <- sensParams(growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff),  nBigMatrix = 50,
minSize = min(dff$size, na.rm=TRUE), maxSize = max(dff$size, na.rm =
TRUE), chosenBin=1, response="lifeExpect")

par(mfrow = c(2, 1), bty = "l", pty = "m")
barplot(resLE$sens,
main = expression("Parameter sensitivity of Life Expectancy"*eta[0]),
las = 2, cex.names = 0.5)
barplot(resLE$elas,
main = expression("Parameter elasticity of Life expectancy"*eta[0]),
las = 2, cex.names = 0.5)

# Same as lambda above, but with two fecundity functions
dff$fec2 <- dff$fec>0 #create binomial describing e.g., prob of flowering
dff$fec[dff$fec==0] <- NA #take out zeros to avoid Inf when fit with log
fv1 <- makeFecObj(dff, Formula = c(fec~size+size2,fec2~size),
    Transform=c("log","none"),Family = c("gaussian","binomial"))

res <- sensParams(growObj=makeGrowthObj(dff), survObj = makeSurvObj(dff),
fecObj = fv1, nBigMatrix = 50, minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE))

par(mfrow = c(2, 1), bty = "l", pty = "m")
barplot(res$sens,
main = expression("Parameter sensitivity of population growth rate " *lambda),
las = 2, cex.names = 0.5)
barplot(res$elas,
main = expression("Parameter elasticity of population growth rate " *lambda),
las = 2, cex.names = 0.5)

# Same but with two fecundity functions and a constant
fv1@fecConstants[1] <-0.5
res <- sensParams(growObj = makeGrowthObj(dff), survObj = makeSurvObj(dff),
fecObj = fv1, nBigMatrix = 50, minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE))

par(mfrow = c(2, 1), bty = "l", pty = "m")
barplot(res$sens,
main = expression("Parameter sensitivity of population growth rate " *lambda),
las = 2, cex.names = 0.5)
barplot(res$elas,
main = expression("Parameter elasticity of population growth rate " *lambda),
las = 2, cex.names = 0.5)

# Same but with a discrete class
dff <- generateData(type="discrete")
res <- sensParams(growObj = makeGrowthObj(dff), survObj = makeSurvObj(dff),
fecObj = makeFecObj(dff), discreteTrans=makeDiscreteTrans(dff),
nBigMatrix = 50, minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE))

par(mfrow = c(2, 1), bty = "l", pty = "m")
barplot(res$sens,
main = expression("Parameter sensitivity of population growth rate " *lambda),
las = 2, cex.names = 0.5)
barplot(res$elas,
main = expression("Parameter elasticity of population growth rate " *lambda),
las = 2, cex.names = 0.5)
```

simulateCarlina *Generates random data in the form used by IPMpack based on the population dynamics of Carlina vulgaris*

Description

Simulates growth, survival and fecundity and density dependent seedling establishment to create a dataframe of the form required by the functions and methods used in IPMpack. Demographic stage data is only continuous. Note that the number or rows corresponding to each year of the data-frame does not inform about population size, since rows exist that correspond to offspring appearing in the subsequent year.

Usage

```
simulateCarlina(nSamp=200,nYrs=1000,nSampleYrs=15,
m0=-1.37,ms=0.59,
b0=-12.05,bs=3.64,
A=-1,B=2,
ag=1.14,bg=0.74,sig=0.29,
mean.kids=3.16,sd.kids=0.5,
meanYear=c(0,0,0),
matVarYear=matrix(c(1.03,0,0,0,0.037,0.041,0,0.041,0.075),3,3),
varA=0,varB=0,densDep=TRUE,
maxPerYr=1000,maxStoreSeedlingsPerYr=200,
sizes = c())
```

Arguments

`nSamp`	number of samples desired in the base population, defaults to 2000
`nYrs`	number of years in the simulation, defaults to 1000
`nSampleYrs`	number of years sampled, defaults to 15
`m0`	intercept survival
`ms`	slope survival
`b0`	intercept flowering
`bs`	slope flowering
`A`	intercept reproductive allometry seed production
`B`	slope reproductive allometry seed production
`ag`	intercept growth
`bg`	slope growth
`sig`	variance growth
`mean.kids`	mean kid size
`sd.kids`	variance kid size
`meanYear`	mean year effects
`matVarYear`	var-covariance in year effects for survival, growth and offspring size

`varA`	variance in seed intercept year effects - defaults to zero
`varB`	variance in seed slope year effects - defaults to zero
`densDep`	density dependence in seedling establishment or not?
`maxPerYr`	total number of individuals for which measurements will be transferred to the subsequent year (population will be resampled with replacement to obtain a population of this size)
`maxStoreSeedlingsPerYr`	max number of seedling recruits for which data will be stored in every year
`sizes`	starting sizes in the population (optional)

Value

A list including: dataf: A dataframe with headings: - "size": continuous variable, indicating current size. - "sizeNext" continuous variable, indicating size in the next time step. - "surv": boolean, indicating whether individual survived or not to the next time step. - "covariate": discrete covariate. - "covariateNext": discrete covariate in the next time step. - "fec": continuous variable, indicating fecundity. - nSeedlings: number seedlings corresponding to that year - m.year: intercept of mortality for that year - cg.year: intercept of growth for that year - b.year: intercept of offspring size for that year - offspringNext: where the row corresponds to offspring, this takes the value offspringNexxt - year: year of the sample

list.par: - a list of all the other parameters matVarYear - variance covariance matrix for demographic functions trueGrow - stochastic growth rate, log lambda s meantrueGrow - mean of lambda t vartrueGrow - variance of log lambda t

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

See Also

`generateData`

Examples

```
#Uncomment to run
#dff <- simulateCarlina(nSamp=1000)
#head(dff$dataf)
```

sizeToAge *Estimates size/stage to age relationships*

Description

Uses a P matrix and a starting continuous stage value to estimate time to a range of target sizes, from which a size to age pattern can be determined.

Usage

```
sizeToAge(Pmatrix, startingSize, targetSize)
```

Arguments

`Pmatrix`	object of class IPMmatrix describing growth survival transitions.
`startingSize`	numeric, size at age 1, or size from which age-size relationship is desired.
`targetSize`	vector of size(s) for which first passage time is required.

Value

`timeInYears`	Time taken to reach target sizes in unit of the time step of the model.
`targetSize`	Vector of target sizes.
`startingSize`	Size at age 1.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Caswell, 2001. Matrix population models: analysis, construction and interpretation. 2nd ed. Sinauer. p110-132.

Metcalf, Horvitz, Tuljapurkar & Clark. 2009 A time to grow and a time to die: a new way to analyze the dynamics of size, light, age and death of tropical trees. Ecology 90, p2766-2778.

Cochran & Ellner. 1992. Simple methods for calculating age-based life history parameters for stage-structured populations. Ecological Monographs 62, p345-364.

See for bias in this estimation where variance in growth is small relative to the size range: Zuidema, Jongejans, Chien, During & Schieving. 2010. Integral Projection Models for trees: a new parameterization method and a validation of model output. Journal of Ecology 98, p345-355.

See Also

`passageTime`

Examples

```
dff <- generateData()
Pmatrix <- makeIPMPmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE), growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff))
targetSize <- 8
startingSize <- 0
sizeToAge(Pmatrix, startingSize, targetSize)
```

`stochGrowthRateManyCov` *Estimates stochastic population growth rates (lambda_s) or invasion rate with many varying covariates. Alternatively, tracks population structure in a stochastic environment if trackStruct is TRUE.*

Description

Iterates a population vector through a time series of covariates according to growth, survival and fecundity objects, and calculates the stochastic population rate of increase if no density-dependence is specified, or the rate of invasion if density dependence is specified.

Usage

```
stochGrowthRateManyCov(covariate, nRunIn, tMax,
growthObj, survObj, fecObj, nBigMatrix, minSize, maxSize,
nMicrosites,integrateType = "midpoint", correction = "none",
trackStruct=FALSE, plot=FALSE, ...)
```

Arguments

`covariate`	matrix with tMax rows, and as many columns as there are relevant covariates.
`nRunIn`	numeric, number of initial samples to discard.
`tMax`	numeric, total number of time-steps to run (same as ncol(covariate)).
`growthObj`	a growth object, defined to correspond to covariate definition (indexing used to make the growth object must match up).
`survObj`	a survival object, defined to correspond to covariate definition in covariate.
`fecObj`	a fecundity object, defined to correspond to covariate definition in covariate.
`nBigMatrix`	numeric, number of size bins in the IPM.
`minSize`	numeric, minimum size in the IPM.
`maxSize`	numeric, maximum size in the IPM.
`nMicrosites`	vector, if sum(nMicrosites)> 0 then density dependence is assumed to operate on seedling establishment, and if length(nMicrosites)>1, then the number of microsites available for establishment at time t is nMicrosites[min(t,length(nMicrosites))].

`integrateType`	integration type, defaults to "midpoint" (which uses probability density function); other option is "cumul" (which uses the cumulative density function).
`correction`	correction type, defaults to "none"; option is "constant" which will multiply every column of the IPM by a constant sufficient to adjust values to those predicted for survival at that continuous stage value.
`trackStruct`	Boolean indicating whether you want to track the population structure (beyond simply estimating the growth rate)
`plot`	Boolean indicating whether a plot of the population structure is desired.
`...`	extra arguments relating to plotting if trackStruct and plot are TRUE.

Details

Forms of density dependence beyond density dependence in seedling establishment not yet defined.

Value

`Rt`	If trackStruct is TRUE, numeric, converging on log lambda_s, or invasion rate (density dependence) for large enough tMax, and if covariate distribution is stationary.
`rc`	matrix of the numbers of individuals in each size and seed class (row) over time (columns).
`IPM.here`	IPM constructed corresponding to pop structure and covariates at tMax.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Ellner & Rees. 2007. Stochastic stable population growth in integral projection models: theory and application. Journal of Mathematical Biology 54, p227-256.

Rees & Ellner. 2009. Integral projection models for populations in temporally varying environments. Ecological Monographs 79, p575-594.

See Also

`stochGrowthRateSampleList`,

Examples

```
### NOT RUN - this is hashed out because compiles too slowly ###

# Generate relevant data, build objects
#dff <- generateData(type="stochastic")
#print(head(dff))
#gr1 <- makeGrowthObj(dff, Formula = sizeNext~size+size2+covariate1)
```

```
#sv1 <- makeSurvObj(dff, Formula = surv~size+size2+covariate2)
#fv1 <- makeFecObj(dff, Formula = fec~size+size2,Transform="log")

# Generate time series of covariates for which population growth rate
#is required. Here set to be seasonal environment.
#Names of covariates must be same as in dff
#tVals <- seq(1,100,by = 1/12)
#covTest <- (1 + 0.5*sin(2*pi*tVals))
#covMatTest <- data.frame(covariate1 = rnorm(length(covTest),covTest,0.5) - 1,
#covariate2 = rnorm(length(covTest), covTest,0.5) - 1,
#covariate3 = rnorm(length(covTest), covTest,0.5) - 1, row.names = NULL)

# Calculate

#r <- stochGrowthRateManyCov(covariate = covMatTest, nRunIn = 5*10,
#tMax = length(tVals), growthObj = gr1, survObj = sv1, fecObj = fv1,
#nBigMatrix = 100,
#minSize = 1.1*min(dff$size, na.rm = TRUE),
#maxSize = 1.1*max(dff$size, na.rm = TRUE), nMicrosites = 0)

#r

# Track population strucuture instead
#st <- stochGrowthRateManyCov(covariate = covMatTest, nRunIn = 5*10,
#tMax = length(tVals), growthObj = gr1, survObj = sv1, fecObj = fv1,
#nBigMatrix = 100,
#minSize = 1.1*min(dff$size, na.rm = TRUE),
#maxSize = 1.1*max(dff$size, na.rm = TRUE), nMicrosites = 0,
#trackStruct=TRUE,plot=TRUE)
```

`stochGrowthRateSampleList` *Estimating the stochastic population growth rate (lambda_s) or invasion rat.*

Description

Estimates the stochastic growth rate (lambda_s) by iteration; operates by sampling a list of IPMs. Note that the function stoch.growth.rate in the package popbio does this more efficiently and with more useful output; but may fail for large bin numbers. If densDept is TRUE, estimates the stochastic invasion rate in the presence of density dependence in seedling establishment by iteration; operates by sampling a list of IPMs and recalculating the probability of seed establishment at each time-step.

Usage

```
stochGrowthRateSampleList(nRunIn,tMax,listIPMmatrix=NULL,
listPmatrix=NULL, listFmatrix=NULL,seedList=NULL,
densDep=FALSE)
```

Arguments

`nRunIn`	numeric, the size of the burnin,
`tMax`	numeric, the total samples desired.
`listIPMmatrix`	a list of IPMmatrix objects corresponding to possible states of the environment.
`listPmatrix`	a list of IPM P matrices corresponding to possible states of the environment.
`listFmatrix`	a list of IPM F matrices corresponding to possible states of the environment.
`seedList`	numeric, a vector of the number of successful recruit corresponding to possible states of the environment.
`densDep`	Boolean indicating whether density dependence in seedling establishment should be implemented

Value

a numeric converging on high enough log lambda_s for sufficient tMax; note that if the population size declines to zero, this may return NAs because of logging.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Caswell, 2001. Matrix population models: analysis, construction and interpretation. 2nd ed. Sinauer. p452-502.

See Also

`sampleSequentialIPMs`

Examples

```
dff <- generateData()
IPMlist <- sampleSequentialIPMs(dataf = dff, nBigMatrix = 10, minSize = -5,
maxSize = 15,fecConstants=data.frame(1e6), correction="constant")
stochGrowthRateSampleList(listIPMmatrix = IPMlist,nRunIn = 100, tMax = 5000)
```

`stochPassageTime` *Estimates passage time in a discretely varying environment.*

Description

Estimates the time in units of the chosen time steps (see convertIncrement()) that it will take to reach a chosen continuous stage value for the first time conditional on surviving from each of the meshpoints of the IPM for each starting environment.

Usage

```
stochPassageTime(chosenSize, IPMmatrix, envMatrix)
```

Arguments

`chosenSize`	numeric, target size for which passage time is desired.
`IPMmatrix`	object of class IPMmatrix describing survival related transitions only.
`envMatrix`	object of class envMatrix describing transitions between discrete environmental states.

Details

Passage time for values exactly equal to the target size are one year, because of how the conditionals are framed. Values slightly less than the target size may on average take longer due to variance in growth, mortality, leading to discontinuities in the pattern of passage time over age. Passage time from values > than the chosenSize should be ignored (space to the right of the red vertical line in example below).

Value

vector containing passage times across size in the first discrete environmental state, then in the second, etc.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez & Eelke Jongejans

References

Caswell, 2001. Matrix population models: analysis, construction and interpretation. 2nd ed. Sinauer.

Metcalf, Horvitz, Tuljapurkar, Clark. 2009. A time to grow and a time to die: a new way to analyze the dynamics of size, light, age and death of tropical trees. Ecology 90, p2766-2778.

For bias in this estimation where variance in growth is small relative to the size range: Zuidema, Jongejans, Chien, During & Schieving. 2010. Integral Projection Models for trees: a new parameterization method and a validation of model output. Journal of Ecology 98, p345-355.

See Also

`passageTime`

Examples

```
dff <- generateData()
Pmatrix <- makeCompoundPmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize=max(dff$size, na.rm = TRUE),
growObj = makeGrowthObj(dff, Formula=sizeNext~size+covariate),
survObj = makeSurvObj(dff, Formula = surv~size+covariate),
envMatrix = makeEnvObj(dff), correction="constant")
```

```
targetSize <- 8

passage <- stochPassageTime(targetSize, Pmatrix, makeEnvObj(dff))

plot(Pmatrix@meshpoints,passage[1:length(Pmatrix@meshpoints)],
ylab = "Passage time", xlab = "Continuous (e.g. size) stage",
type = "l", col = "dark gray",
xlim=c(Pmatrix@meshpoints[1],targetSize),
ylim = c(0, max(passage)))
abline(v = targetSize, col = "red")
points(Pmatrix@meshpoints,
passage[(length(Pmatrix@meshpoints)+1):(2*length(Pmatrix@meshpoints))],
type="l",col="green")
```

surv *Survival*

Description

Predicts the probability of surviving at a given size given a survival object.

Usage

```
surv(size, cov, survObj)
```

Arguments

`size`	a numeric vector of current sizes.
`cov`	a data-frame with one row containing all covariates.
`survObj`	a survObj.

Value

a vector of length size with values between 0 and 1.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

See Also

`growth`, `growSurv`, `surv-methods`

Examples

```
dff <- generateData()
sv1 <- makeSurvObj(dff)
surv(1:50, data.frame(cov=1), sv1)
```

`surv-methods` *~~ Methods for Function surv ~~*

Description

~~ Methods for function `surv` ~~

Methods

`signature(size = "numeric", cov = "data.frame", survObj = "survObj")` Methods to predict probability of survival given a linear predictor based on various transforms of size and covariates defined in cov.

`signature(size = "numeric", cov = "data.frame", survObj = "survObjOverDisp")` Methods to predict probability of survival given a linear predictor based on various transforms of size and acovariate, where over-dispersion has been fitted, using a correction.

`survivorship` *Estimates survivorship between two time censuses.*

Description

Calculates the fraction of the cohort surviving across age for a chosen starting continuous stage value.

Usage

```
survivorship(IPMmatrix, loc, maxAge)
```

Arguments

`IPMmatrix`	an IPMmatrix object describing growth and survival transitions across stage (e.g. size) and environment.
`loc`	a starting size location in the IPM matrix for age 1 (i.e., either the index of the desired size in the meshpoints, or, if there are discrete stages, the index + the number of discrete stages; if this is not an integer, then it will be assumed that the rounded version is desired)
`maxAge`	the maximum age up to which survivorship is desired for or possible.

Value

`surv.curv`	vector of length maxAge providing survivorship at each age from 1 to maxAge.
`stageAgeSurv`	matrix of dimensions nBigMatrix*maxAge providing the population structure at every age for a cohort starting with an individual of size size1.
`mortality`	vector of length maxAge providing mortality at each age from 1 to maxAge.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Tuljapurkar & Horvitz. 2006. From stage to age in variable environments. Life expectancy and survivorship. Ecology 87, p1497-1509.

Examples

```
# For only continuous stages (e.g. size)
dff <- generateData()
Pmatrix <- makeIPMPmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE), growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff))
su <- survivorship(Pmatrix, 1, 300)
plot(su$surv.curv, type = "l", col = "dark gray", ylab = "survivorship",
xlab= "Continuous (e.g. size) stage", ylim = c(0,1))

# For continuous (e.g. size) and discrete (e.g. seedbank) stages
Pmatrix <- makeCompoundPmatrix(minSize = min(dff$size,na.rm = TRUE),
maxSize = max(dff$size,na.rm = TRUE), envMatrix = makeEnvObj(dff),
growObj = makeGrowthObj(dff, Formula = sizeNext~size+size2+covariate),
survObj = makeSurvObj(dff, Formula = surv~size+size2+covariate),
discreteTrans = 1)
su <- survivorship(Pmatrix,1,300)
```

`survObj-class` *Class "survObj"*

Description

A class object description

Objects from the Class

Objects can be created by calls of the form `new("survObj", ...)`.

Slots

`fit`: Object of class `"glm"` ~~

Methods

surv `signature(size = "numeric", cov = "numeric", survObj = "survObj")`: ...

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez & Eelke Jongejans

Examples

```
showClass("survObj")
```

---

survObjOverDisp-class *Class* "survObjOverDisp"

Description

A class object description

Objects from the Class

Objects can be created by calls of the form `new("survObjOverDisp", ...)`.

Slots

`fit`: Object of class `"glm"` ~~

Methods

surv `signature(size = "numeric", cov = "numeric", survObj = "survObjOverDisp")`: ...

Examples

```
showClass("survObjOverDisp")
```

---

timeToSize *Projects how long it takes to get from a starting distribution to a target continuous stage value.*

Description

Provided with a starting vector reflecting starting individual sizes, this function projects forward via the provided IPM until a defined proportion of the population has reach the chosen endSize. Only works for single environment or compound matrices (not time-varying covariates apart from a single discrete one).

Usage

```
timeToSize(startingSizes, IPM, endSize, startingEnv = 1, maxT = 100, propReach = 0.01)
```

Arguments

`startingSizes`	vector of starting sizes reflecting sizes of individuals in the starting population (in any order).
`IPM`	the IPM Pmatrix.
`endSize`	the end size.
`startingEnv`	vector of starting env, same length as startingSizes, or length = 1 if compound matrices are not being used.
`maxT`	the max number of time steps tested.
`propReach`	the proportion of the starting pop that have to be > than the endSize for it to count.

Details

Plots and returned values of survivorship from preliminary runs will give a notion of how low this has to be.

Value

`ts.dist`	the time-series of size distribution
`time.reach`	the time for n.reach to be at sizes > endSize
`survivorship`	survivorship over the course of the time elapsed for that pop

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Caswell, 2001. Matrix population models: analysis, construction and interpretation. 2nd ed. Sinauer.

Examples

```
#note that with the "fake data" essentially either takes
#forever or is immediate...
dff <- generateData()
startSizes <- rnorm(1000, 2.5, 1)
Pmatrix <- makeIPMPmatrix(minSize = 1.2*min(dff$size, na.rm=TRUE),
maxSize = 1.2*max(dff$size, na.rm=TRUE),
growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff))

rc <- timeToSize(startingSizes = startSizes, IPM = Pmatrix, endSize = 6,
startingEnv = 1, maxT = 1000, propReach = 0.001)

names(rc)
```

```
par(mfrow=c(2,2), bty = "l")
## Make picture with lines for distribution of
## population on different time points
matplot(Pmatrix@meshpoints, rc$ts.dist, type = "l", xlab = "size",
ylab = "Number of individuals")

## Examine time elapsed for propReach to attain the chosen endSize
rc$time.reach

## Plot out the survivorship
plot(rc$survivorship, type = "l", #log = "y",
xlab = "time step", ylab = "Probability original population survival",
ylim = c(0,1), col = "gray")
```

---

varLifeExpect — *Calculates variation in life expectancy in a discretely stochastic environment.*

Description

Provided a P matrix, defining survival transitions across size, this function provides a vector with variance in life expectancy in units of the time-step used, for each of the size bins.

Usage

```
varLifeExpect(IPMmatrix)
```

Arguments

`IPMmatrix`	an IPMmatrix object defining survival transitions.

Value

a vector of variance in life expectancies each corresponding to Pmatrix@meshpoints.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Caswell, 2001. Matrix population models: analysis, construction and interpretation. 2nd ed. Sinauer. p110-132.

Cochran & Ellner. 1995. Simple methods for calculating age-based life history parameters for stage-structured populations. Ecological Monographs 62, p345-364.

Tuljapurkar & Horvitz, 2006. From stage to age in variable environments. Life expectancy and survivorship. Ecology 87, p1497-1509.

See Also

meanLifeExpect, makeIPMPmatrix

Examples

```
# With a single continuous (e.g. size) stage
dff <- generateData()
Pmatrix <- makeIPMPmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize = max(dff$size, na.rm = TRUE), growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff))
vle <- varLifeExpect(Pmatrix)

plot(Pmatrix@meshpoints, vle, ylab = "Variation life expectancy",
xlab = "Continuous (e.g. size) stage", type = "l", ylim = c(0,max(vle)))
```

`varPassageTime` *Estimates variation in passage time.*

Description

Function to take a P matrix (either compound or not) and estimate variance in passage time to a chosen continuous stage value.

Usage

```
varPassageTime(chosenSize, IPMmatrix)
```

Arguments

chosenSize	The continuous stage value of interest.
IPMmatrix	The Pmatrix (compound or not).

Details

Note how variation in passage time for values exactly equal to the chosen size (targetSize) are low, because of way the conditionals are framed. Passage time from values > than targetSize should be ignored (space to the right of the red vertical line in example below), unless dealing with an organism that is able to display retrogression.

Value

Numeric vector corresponding to variance in passage time from each of the meshpoints in the IPM (so both size, and if a compound matrix, size from different environments).

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Caswell, 2001. Matrix population models: analysis, construction and interpretation. 2nd ed. Sinauer. p119.

Metcalf, Horvitz, Tuljapurkar & Clark. 2009. A time to grow and a time to die: a new way to analyze the dynamics of size, light, age and death of tropical trees. Ecology 90, p2766-2778.

For bias in this estimation where variance in growth is small relative to the size range: Zuidema, Jongejans, Chien, During & Schieving. 2010. Integral Projection Models for trees: a new parameterization method and a validation of model output. Journal of Ecology 98, p345-355.

For species with shrinkage: Salguero-Gomez & Casper. 2010. Keeping shrinkage in the demographic loop. Journal of Ecology 98, p313-323.

See Also

`passageTime`, `makeIPMPmatrix`

Examples

```
# With continuous (e.g. size) stage
dff <- generateData()
Pmatrix <- makeIPMPmatrix(minSize = min(dff$size, na.rm = TRUE),
maxSize <- max(dff$size, na.rm = TRUE), growObj = makeGrowthObj(dff),
survObj = makeSurvObj(dff))
targetSize <- 8
vP <- varPassageTime(targetSize, Pmatrix)

plot(Pmatrix@meshpoints, vP, type = "l", xlab="Continuous (e.g. Size) stage",
xlim=c(Pmatrix@meshpoints[1],targetSize),
ylab = "Variance in passage time", col = "dark gray")
abline(v = targetSize, col = "red")

## not sure variance works with this....
```

`wrapHossfeld` *Fitting Hossfeld growth function.*

Description

Function to define deviance of a Hossfeld function for use with optim to generate a Hossfeld growth object.

Usage

```
wrapHossfeld(par, dataf)
```

Arguments

`par`	vector of length three.
`dataf`	dataframe with columns size and incr.

Author(s)

C. Jessica E. Metcalf, Sean M. McMahon, Roberto Salguero-Gomez, Eelke Jongejans & Cory Merow.

References

Zuidema, Jongejans, Chien, During & Schieving. Integral projection models for trees: a new parameterization method and a validation of model output. Journal of Ecology 98, p345-355.

See Also

`Hossfeld`

Examples

```
# Simulate data and create a column for growth increment
dff <- generateData()
dff$incr <- dff$sizeNext - dff$size

# Current sum of squares
wrapHossfeld(c(1, 1, 1), dff)

# Use optim to get best parameters values [not run]
tmp <- optim(c(1, 1, 1), wrapHossfeld, dataf = dff, method = "Nelder-Mead")
```

Index