Package ‘primer’

October 14, 2022

Type Package
Title Functions and Data for the Book, a Primer of Ecology with R
Version 1.2.0
Date 2021-01-06
Author M H H Stevens
Maintainer Hank Stevens <hank.stevens@miamioh.edu>
Depends deSolve, ggplot2, R (>= 3.10)
Imports tidyR, graphics, stats, utils
Suggests bbmle, bipartite, cowplot, data.table, diagram, DiagrammeR,
         DiagrammeRsvg, gdata, igraph, lavaan, magrittr, nlme, phaseR,
         vegan, reshape2, rARPACK, rsvg
Description Functions are primarily functions for systems of ordinary differential equations, difference
         equations, and eigenanalysis and projection of demographic matrices; data are for examples.
License GPL-3
LazyLoad yes
NeedsCompilation no
RoxygenNote 7.1.1
Encoding UTF-8
Repository CRAN
Date/Publication 2021-01-06 21:10:02 UTC

R topics documented:

primr-package ......................................................... 2
allogistic ........................................................... 3
bip_stability .......................................................... 4
BSsucc ................................................................. 6
CandG ................................................................. 7
cat_function .......................................................... 7
chesson ................................................................. 8
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>clogistic</td>
<td>9</td>
</tr>
<tr>
<td>ClostExp</td>
<td>11</td>
</tr>
<tr>
<td>compcol</td>
<td>11</td>
</tr>
<tr>
<td>compcolM</td>
<td>12</td>
</tr>
<tr>
<td>coneflower</td>
<td>13</td>
</tr>
<tr>
<td>coneflowerrecruits</td>
<td>14</td>
</tr>
<tr>
<td>coneflowerseeds</td>
<td>15</td>
</tr>
<tr>
<td>dlogistic</td>
<td>15</td>
</tr>
<tr>
<td>dlvcomp2</td>
<td>16</td>
</tr>
<tr>
<td>gotelli</td>
<td>17</td>
</tr>
<tr>
<td>hanski</td>
<td>18</td>
</tr>
<tr>
<td>igp</td>
<td>19</td>
</tr>
<tr>
<td>lande</td>
<td>20</td>
</tr>
<tr>
<td>levins</td>
<td>22</td>
</tr>
<tr>
<td>lvcomp2</td>
<td>23</td>
</tr>
<tr>
<td>lvcomp3</td>
<td>24</td>
</tr>
<tr>
<td>lvcompg</td>
<td>25</td>
</tr>
<tr>
<td>LVCompGames</td>
<td>26</td>
</tr>
<tr>
<td>MetaSim</td>
<td>28</td>
</tr>
<tr>
<td>moths</td>
<td>29</td>
</tr>
<tr>
<td>one_over_f</td>
<td>30</td>
</tr>
<tr>
<td>pimmlawton</td>
<td>31</td>
</tr>
<tr>
<td>plot_f</td>
<td>33</td>
</tr>
<tr>
<td>PopSim</td>
<td>34</td>
</tr>
<tr>
<td>predpreyLV</td>
<td>35</td>
</tr>
<tr>
<td>predpreyRM</td>
<td>36</td>
</tr>
<tr>
<td>ross</td>
<td>37</td>
</tr>
<tr>
<td>scheffer</td>
<td>37</td>
</tr>
<tr>
<td>SIRbd</td>
<td>38</td>
</tr>
<tr>
<td>SIRd</td>
<td>38</td>
</tr>
<tr>
<td>SIRf</td>
<td>40</td>
</tr>
<tr>
<td>SIRmod</td>
<td>41</td>
</tr>
<tr>
<td>sparrows</td>
<td>43</td>
</tr>
<tr>
<td>spec_mimic</td>
<td>44</td>
</tr>
<tr>
<td>succniche</td>
<td>45</td>
</tr>
<tr>
<td>thetalogistic</td>
<td>46</td>
</tr>
<tr>
<td>weeds</td>
<td>47</td>
</tr>
</tbody>
</table>

**Index**

<table>
<thead>
<tr>
<th>primer-package</th>
<th>Functions and data for &quot;A Primer of Ecology with R&quot;</th>
</tr>
</thead>
</table>

**Description**

Functions are primarily functions for systems of ordinary differential equations, difference equations, and eigenanalysis and projection of demographic matrices; data are for examples.
**alogistic**

**Details**

Package: primer  
Type: Package  
Version: 1.0  
Date: 2012-05-16  
License: GPL version 2 or later  
LazyLoad: yes

**Author(s)**

Hank Stevens <HStevens@muohio.edu>

**References**


**See Also**

`lvcompg`, `ode`

---

| alogistic          | Logistic growth with the Allee effect |

**Description**

A function for continuous logistic growth with the Allee effect, for use with `ode` in the `deSolve` package.

**Usage**

`alogistic(t, y, p)`

**Arguments**

- **t**  
  Time points for which `N` will be returned.
- **y**  
  `N`, population size
- **p**  
  a vector of logistic growth (`r`, `alpha`) and Allee effect parameters (`a`, `tau`); `a` is the threshold population size.
Details

The user does not put these directly into this function, but rather uses code in the deSolve package. The function is based on the logistic growth equation

\[ \frac{dN}{dt} = rN \left(1 - \alpha N \right) \left(1 - \frac{a + \tau}{N + \tau} \right) \]

with \( \alpha \) being the threshold population size.

Author(s)

Hank Stevens

References


See Also

clogistic, dlogistic, thetalogistic, lewins

Examples

```r
library(deSolve)
p <- c(r=1, alpha=.01, a=10, tau=.1)
time <- 0:10
initialN <- 11
out <- ode(y=initialN, times=time,
func=alogistic, parms=p)
plot(time, out[-1], type='l')
```

bip_stability

Bipartite network properties

Description

A function to calculate the several properties of randomly constructed bipartite networks. This function installation of packages bipartite, igraph, and rARPACK.

Usage

```r
bip_stability(
Spp.a = 10:50,
Spp.p = 5:25,
C = c(0.05, 0.3),
IS = NULL,
quant = FALSE,
ndd = 1,
reps = 10
)`
Arguments

- **Spp.a**: numeric vector from which to draw a random number of animal species
- **Spp.p**: numeric vector from which to draw a random number of animal species
- **C**: numeric two element vector for minimum and maximum possible connectances.
- **IS**: mean interaction strength. If NULL, it is calculated as the inverse of the matrix dimension, so that the expected sum of interaction strengths equals one.
- **quant**: Logical indicating whether the estimate of modularity should be based on the binary adjacency matrix (FALSE, default), or a quantitative matrix (TRUE).
- **ndd**: scalar (1 or 2) determining whether the intraspecific negative density dependence is the same in both networks (1), or is scaled to the interaction sizes in each matrix (2).
- **reps**: number of random matrices.

Details

The interaction strengths are exponentially distributed, with mean of IS (rate = 1/IS). Nestedness is WNODF (Almeida-Neto et al. 2011). It can calculate modularity in two ways, using either the binary adjacency matrix (ones and zeroes, Clauset, Newman and Moore 2004) or a quantitative matrix (Beckett 2016); the latter takes much more time to calculate, so it uses the binary matrix by default. It also tallies diversity as the total number of species and the connectance as the total number of links divided by the number of possible links.

To calculate resilience, the P by A bipartite matrix is expanded to a (P+A) x (P+A) symmetric matrix so the upper triangular portion are the effects of the animals on the plants, and the lower triangular portion is the effects of the plants on the animals.

The diagonal represents intraspecific density dependence; it is filled with the negative of the largest summed row of interaction strengths in the symmetric matrix. The symmetric matrix is interpreted as a Jacobian matrix. Therefore, resilience is the negative value of the real part of the dominant eigen value of the expanded symmetric matrix.

Value

Returns a list of a matrix with columns for species richness, connectance, nestedness, modularity, and resilience.

Author(s)

Hank Stevens <HankStevens@miamioh.edu>

References

**BSsucc**

*Secondary succession data*

---

**Description**

Functional group abundances (herbaceous annual, herbaceous perennial, woody) from one of the fields in the Buell-Small long term succession study (http://www.ecostudies.org/bss/). Data are based on visual estimates of percent cover, using annual means of each species, which are then summed for each functional group.

**Format**

A data frame with 147 observations on the following 3 variables.

- **AGE** a numeric vector; indicates the age of succession since abandonment from agriculture.
- **variable** plant functional type; a factor with levels `Annual`, `Perennial`, `Woody`
- **value** a numeric vector

**Source**

Data graciously provided by Scott Meiners (Eastern Illinois University, http://www.ecostudies.org/bss/).

**References**


**Examples**

```r
# b <- bip_stability(Spp.a = 16:60, Spp.p = 8:30, C=c(.05, 0.5), reps=10, quant=FALSE)
# pairs(b)
```

```r
data(BSsucc)
lattice::xyplot(value ~ AGE, groups=variable, data=BSsucc, 
    type='smooth', span=.3, ylab="Percent Cover", 
    xlab="Age Since Abandonment (y)", 
    auto.key=list(columns=3, space="top", lines=TRUE, 
    points=FALSE))
```
CandG

Data drawn approximately from Collins and Glenn (1991)

Description

Numbers of species which were observed in 1–19 sites.

Format

The format is: num [1:19] 32 16 10 9 8 7 8 6 4 5 ...

Source

Data are approximations of histogram values for a single year of data in:

References


Examples

data(CandG)
barplot(CandG, names=1:19)

cat_function

A Cat Function

Description

This function allows you to express your love of cats.

Usage

cat_function(love = TRUE)

Arguments

love Do you love cats? Defaults to TRUE.

Author(s)

Hilary Parker
Examples

cat_function()

chesson  Two-species model of the storage effect

Description

Simulates a fluctuating environment over time, and two species’ responses to the environment, after Chesson (1994).

Usage

chesson(
  alpha = c(1.1 * 1e-05, 1e-05),
  d = 0.1,
  years = 10,
  N0 = c(1000, 1e+05),
  w = c(0.6, 1),
  env.var = 1,
  specialization = 1,
  spread = 0.67,
  type = 1
)

Arguments

alpha  a vector of length 2; the negative effects of all individuals (of both species) on each population – typically different among species.
d  disturbance rate; the proportion of all individuals killed at each time step.
years  numbers of time steps
N0  vector of length 2; initial abundances.
w  vector of length 2; average fitnesses for each species.
env.var  degree of environmental variability.
specialization  determines the narrowness of each species fitness response.
spread  determines how far apart the peak fitness responses are.
type  determines the form of C, the negative effect of competition.

Details

The argument type controls the value of $e^C$, the effect of competition on reproduction, where the annual finite rate of increase is $R = e^{E-C}$. type = 1 causes $e^C = \alpha_i N_{J_i}$, that is, a species-specific fixed fraction of juveniles that depends on each species response to competition. This is illustrated in a for-loop in Stevens (2009, Ch. 9, Storage Effect, Simulating Dynamics). Any other value for type results in the same negative effect on both species that depends on the number of juveniles and the disturbance rate.
clogistic

Value

Returns a list of length one, for use with ode in the deSolve package.

Component 1 vector of the state variables, y.

Author(s)

Hank Stevens (HankStevens@miamioh.edu)

References


See Also

succniche

Examples

```r
out <- chesson(years=50)
layout(matrix(1:4, nc=2))
matplot(out["time"], out["Ns"], type='l', lty=c(1:3),
  xlab="Time", ylab="N", log="y")
plot(out["time"][-1], out["env"], type='l',
  xlab="Time", ylab="Environment")
matplot(out["env"], out["Es"], xlab="Environment",
  ylab="Density-independent reproduction")
matplot(out["env"], out["Rs"], xlab="Environment",
  ylab="Annual growth rate")
```

clogistic  

*Continuous Logistic Growth*

Description

A function for continuous logistic growth, for use with ode in the deSolve package.

Usage

clogistic(times, y, parms)
Arguments

times		times points that will return N

y		N

parms		a vector of logistic growth parameters; the first must be r, and the second must be alpha (1/K).

Details

The user does not put these directly into this function, but rather uses ode in the deSolve package.

The function implements the logistic growth equation

\[ \frac{dN}{dt} = rN (1 - \alpha N) \]

or equivalently

\[ \frac{dN}{dt} = rN \left( \frac{K - N}{K} \right) \]

Value

Returns of list of one component (required by ode).

Author(s)

Hank Stevens <HankStevens@miamioh.edu>

References


See Also

dlogistic, thetalogistic, levins, lvcomp2, lvcompg

Examples

```r
library(deSolve)
p <- c(r=1, alpha=.01)
time <- 1:10
initialN <- 10
out <- ode(y=initialN, times=time, 
func=clogistic, parms=p)
plot(time, out[, -1], type='l')
```
Description

Data set from an experiment testing whether trophic heterogeneity influences long term population sizes.

Format

A data frame with 144 observations on the following 5 variables.

Nutrients a factor with levels high low
No.per.ml a numeric vector
Day a numeric vector
rep a factor with levels a b c d
ID an ordered factor with levels a.low < d.low < c.low < b.low < c.high < a.high < d.high < b.high

Source


References


Examples

data(ClostExp)
#lattice::xyplot(log10(No.per.ml) ~ Day|Nutrients, data=ClostExp, groups=rep, type='l')

```r
compcol
```

Usage

compcol(t, y, params)
Arguments

- **t**  
  Argument for time

- **y**  
  A vector for population 1 and 2

- **params**  
  Vector or list of parameters

Value

Returns a list of length one, for use with ode in the deSolve package.

Component 1 vector of the state variables, y.

Author(s)

Hank Stevens <HankStevens@miamioh.edu>

References


See Also

- levins
- compcolM
- succniche

Examples

```r
library(deSolve)
pars <- c(c1 = .3, c2 = 1, m1 = .1, m2 = .1)
pops <- c(.1,.1)
out <- ode(y=pops, t=0:20, fun=compcol, parms = pars)
matplot(out[,1], out[,-1], type='l')
```

Description

Multi-species competition Colonization Model, With Habitat Destruction

Usage

```
compcolM(t, y, params)
```
Arguments

- **t**: Argument for each time point
- **y**: A vector for the populations
- **params**: Vector or list of parameters

Value

Returns a list of length one, for use with `ode` in the `deSolve` package.

Component 1 vector of the state variables, y.

Author(s)

Hank Stevens <HStevens@muohio.edu>

References


See Also

levins, compcol, succniche

Examples

```r
library(deSolve)
S <- 10
ci <- 2^seq(-5, 5, length=S)
m <- rep(.1, S)
params <- list(ci=ci, m=m, S=S, D=0)
init.N <- rep(0.01, S); t=seq(1, 200, .1)
cc.out <- ode(init.N, t, compcolM, params)
matplot(t, cc.out[, -1], type="l", ylab="Proportion of Habitat", xlab="Years")
```

---

**coneflower**

*Smooth coneflower data*

Description

A data set containing size, survival, and flowering in smooth coneflower
Usage

data(coneflower)

Format

A data frame with 136 rows and 5 variables:

- **logA**: natural log of total leaf area in the first year
- **logAp**: natural log of total leaf area in the second year (Ap = A prime)
- **surv**: indicator of survival from year one to year 2 (0,1)
- **flower_p**: indicator of flowering in year 2 (0,1)

Source

Data provided graciously by Rachel Collins (Roanoke College).

Examples

```r
  data(coneflower)
  ggplot2::qplot(logA, logAp, data=coneflower)
```

---

**coneflowerrecruits**  
*Smooth coneflower new recruits*

Description

A data set containing recruit size and survival.

Usage

```r
  data(coneflowerrecruits)
```

Format

A data frame with 36 rows and 3 variables:

- **area17**: total leaf area in the first year
- **surv18**: indicator of survival from year one to year 2 (0,1)
- **area19**: total leaf area in the third year
**coneflowerseeds**

*Smooth coneflower seed data*

**Description**

A data set containing size and seed set of individual coneflowers.

**Usage**

```r
data(coneflowerseeds)
```

**Format**

A data frame with 136 rows and 5 variables:

- `logAs` natural log of total leaf area in the first year
- `seeds` number of seeds produced in the first year

**Examples**

```r
data(coneflowerseeds)
ggplot2::qplot(logAs, seeds, data=coneflowerseeds)
```

---

**dlogistic**

*Discrete Logistic Growth*

**Description**

Single species discrete logistic growth – a difference equation. A function for continuous logistic growth, for use with `ode` in the `deSolve` package, using method = 'euler' and integer time steps.

**Usage**

```r
dlogistic(t, y, p)
```

**Arguments**

- `t` times points that will return N
- `y` N
- `p` a vector of labeled logistic growth parameters; the first must be labeled rd, and the second must be labeled alpha (the value of alpha is 1/K).
dlvcomp2

Details
Of the form,

\[ N_{t+1} - N_t = r_d N_t (1 - \alpha N_t) \]

Value
Returns of list of one component (required by ode).

Author(s)
Hank Stevens (HankStevens@miamioh.edu)

References

See Also
clogistic.lvcomp

Examples

library(deSolve)
# MUST use the 'euler' integration method with integer time steps
p <- c(rd=1, alpha=.01)
time <- 0:10
initialN <- 10
out <- ode(y=initialN, times=time,
    func=dlogistic, parms=p, method='euler')
plot(time, out[,,-1], type='l')

---

dlvcomp2 Discrete Logistic Competition

Description
A function to simulate discrete 2 species Lotka-Volterra competition.

Usage
dlvcomp2(N, alpha, rd = c(1, 1))

Arguments
N a vector of length two, containing N[t] for both species.
alpha a 2 x 2 matrix of the magnitudes of the per capita (negative) effects (i.e., positive value results in a negative effect).
rd a vector of length 2 containing the discrete growth increments for 2 species.
**Value**

Returns a vector of length 2, containing N[t+1] for both species.

**Author(s)**

Hank Stevens (HStevens@muohio.edu)

**References**


**See Also**

dlogistic, lvcomp2, lvcomp3, lvcompg

**Examples**

```r
alphs <- matrix(c( .01, .005,
                   .008, .01), ncol=2, byrow=TRUE)
t <- 20
N <- matrix(NA, nrow=t+1, ncol=2)
N[1,] <- c(10,10)
for(i in 1:t) N[i+1,] <- dlvcomp2(N[i,], alphs)
matplot(0:t, N, type='l', col=1, ylim=c(0,110))
abline(h=1/alphs[1,1], lty=3)
text(0, 1/alphs[1,1], "K", adj=c(0,0))
legend("right", c(expression("Sp.1 "*(alpha[1]==0.008)),
               expression("Sp.2 "*(alpha[2]==0.005))), lty=1:2, bty='n')
```

---

**gotelli**

*Propagule Rain Metapopulation Model*

**Description**

A function for the propagule rain or mainland-island metapopulation dynamics, for use with ode in the deSolve package.

**Usage**

gotelli(t, y, parms)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>Argument for time</td>
</tr>
<tr>
<td>y</td>
<td>A scalar for the population variable</td>
</tr>
<tr>
<td>parms</td>
<td>Vector or list of parameters</td>
</tr>
</tbody>
</table>
Value

Returns a list of length one, for use with ode in the deSolve package.

Component 1 vector of the state variable (a scalar for the proportion of sites occupied).

Author(s)

Hank Stevens <HStevens@muohio.edu>

References


See Also

gotelli, hanski, lande, MetaSim, clogistic

Examples

```r
## The function is currently defined as
function (t, y, parms)
{
    p <- y[1]
    with(as.list(parms), {
        dp <- ce * (1 - p) - e * p
        return(list(dp))
    })
}
```

```r
library(deSolve)
p <- c(ce=.1, e=.01)
time <- 1:10
initialN <- .3
out <- ode(y=initialN, times=time, func=gotelli, parms=p)
plot(time, out[,,-1], type='l')
```

---

**hanski**

*Core-satellite Metapopulation Model*

Description

A function for the core-satellite metaapopulation dynamics, for use with ode in the deSolve package.
igp

Usage

hanski(t, y, parms)

Arguments

t Argument for time

y A scalar for the population variable

parms Vector or list of parameters

Value

Returns a list of length one, for use with ode in the deSolve package.

Component 1 vector of the state variable (a scalar for the proportion of sites occupied).

Author(s)

Hank Stevens <HStevens@muohio.edu>

References


See Also

gotelli, hanski, lande, MetaSim, clogistic

Examples

prms <- c(ci<- 0.15, e=0.05)
out <- ode(y=2, times=1:100, func=hanski, parms=prms )
matplot(out[,1], out[,2], type='l', ylab="p", xlab="time")

igp

A Lotka-Volterra Model of Intraguild Predation

Description


Usage

igp(t, y, params)
Arguments

t the time point for a realization of the integration.
y the vector of populations, at each time t.
params a vector or list containing the necessary parameters.

Value

Returns a list of length one which is the vector of the rates of increase (required by ode).

Author(s)

Hank Stevens <HStevens@muohio.edu>

References


See Also

lvdcompg, scheffer

Examples

```r
library(deSolve)
params <- c(bpb= 0.032, abp=10^-8, bpn=10^-5, anp=10^-4, mp=1,
           bnb=0.04, abn=10^-8, mn=1,
r=1, rbb=10^-9.5)
t=seq(0, 60, by=.1)
N.init <- c(B = 10^9, N = 10^4, P = 10^3)
igp.out <- ode(N.init, t, igp, params)
matplot(t, log10(igp.out[-1]+1), type="l",
       ylab="log(Abundance)"
legend('right', c("B", "N", "P"), lty=1:3, col=1:3, lwd=2,
bty="n")
```

lande

*A Metapopulation Model with Habitat Destruction*

Description

A function for the metaapopulation dynamics, for use with ode in the deSolve package.
Usage

\texttt{lande(t, y, parms)}

Arguments

t \quad \text{Argument for time}

y \quad \text{A scalar for the population variable}

parms \quad \text{Vector or list of parameters}

Value

Returns a list of length one, for use with ode in the \texttt{deSolve} package.

Component 1 \quad \text{vector of the state variable (a scalar for the proportion of sites occupied).}

Author(s)

Hank Stevens \textless{}HStevens@muohio.edu\textgreater{}

References


See Also

gotelli, hanski.levins, MetaSim, clogistic

Examples

```r
## The function is currently defined as
function (t, y, parms)
{
  p <- y[1]
  with(as.list(parms), {
    dp <- ci * p * (1 - D - p) - e * p
    return(list(dp))
  })
}

library(deSolve)
p <- c(ci=.1, e=.01, D=.5)
time <- 1:10
initialN <- .3
out <- ode(y=initialN, times=time, func=lande, parms=p)
plot(time, out[,,-1], type='l')
```
levins  Classic Metapopulation Model

Description

A function for the classic metapopulation dynamics, for use with ode in the deSolve package.

Usage

levins(t, y, parms)

Arguments

t  Argument for time
y  A scalar for the population variable
parms  Vector or list of parameters

Value

Returns a list of length one, for use with ode in the deSolve package.

Component 1  vector of the state variable (a scalar for the proportion of sites occupied).

Author(s)

Hank Stevens <HStevens@muohio.edu>

References


See Also

gotelli, hansi, lande, MetaSim, clogistic

Examples

```r
# The function is currently defined as
function (t, y, parms)
{
  p <- y[1]
  with(as.list(parms), {
    dp <- ci * p * (1 - p) - e * p
    return(list(dp))
  })
} 
```
lvcomp2

Two Species Lotka-Volterra Competition

Description

System of ordinary differential equations for two species Lotka-Volterra competition. For use with ode in the deSolve package.

Usage

lvcomp2(t, n, parms)

Arguments

t Times points that will return N.
n a vector of length two for the population sizes at time = t.
parms vector or list of model parameters (see details below).

Details

The parameters include r1, r2, a11, a12, a21, a22 with the usual meanings. Here the a’s are the per capita effects which determine K (a11 = 1/K).

Value

Returns a list of length one which is the rate of increase (required by ode).

Author(s)

Hank Stevens <HStevens@muohio.edu>

References


See Also

lvcomp3, lvcompg, clogistic
Examples

```r
## The function is currently defined as
function (t, n, parms)
{
    with(as.list(parms), {
        dn1dt <- r1 * n[1] * (1 - a11 * n[1] - a12 * n[2])
        list(c(dn1dt, dn2dt))
    })
}
```

```r
library(deSolve)
parms <- c(r1 = 1, r2 = 0.1, a11 = 0.2, a21 = 0.1, a22 = 0.02, a12 = 0.01)
initialN <- c(1, 1)
out <- ode(y = initialN, times = 1:100, func = lvcomp2, parms = parms)
matplot(out[, 1], out[, -1], type = "l")
```

---

### lvcomp3

**Three Species Lotka-Volterra Competition**

**Description**

System of ordinary differential equations for three species Lotka-Volterra competition. For use with `ode` in the `deSolve` package.

**Usage**

```r
lvcomp3(t, n, parms)
```

**Arguments**

- `t` the time for each integration.
- `n` a vector of length three for the population sizes at time = t.
- `parms` vector or list of model parameters (see details below).

**Details**

The parameters include `r`, `a` with the usual meanings. Here the `a`'s are the per capita effects which determine `K` (`a11 = 1/K1`).

**Value**

Returns a list of length one which is the rate of increase (required by `ode`).

**Author(s)**

Hank Stevens <HStevens@muohio.edu>
References

See Also
*lvcomp2*, *lvcompg*, *clogistic*

Examples

```r
## The function is currently defined as
function (t, n, parms)
{
  with(as.list(parms), {
    list(c(dn1dt, dn2dt, dn3dt))
  })
}

library(deSolve)
parms <- c(r1 = 0.1, r2 = 0.2, r3 = 0.3,
a11 = 0.1, a12 = 0.01, a13 = 0.01,
a21 = 0.01, a22 = 0.15, a23 = 0.01,
a31 = 0.01, a32 = 0.01, a33 = 0.2)
initialN <- c(1, 1, 1)
out <- ode(y = initialN, times = 1:100, func = lvcomp3, parms = parms)
matplot(out[, 1], out[, -1], type = "l")
```

**lvcompg**

*A General Lotka-Volterra Competition Model*

Description
A general Lotka-Volterra competition model, for any number of species. For use with ode in the deSolve package. This function uses a vector and matrix within the list of parameters.

Usage

`lvcompg(t, n, parms)`
Arguments

- **t**
  - the time point for a realization of the integration.
- **n**
  - the vector of populations, at each time t.
- **parms**
  - a LIST containing a vector of growth rates (r), and a matrix of interaction coefficients (a).

Value

Returns a list of length one which is the vector of the rates of increase (required by ode).

Author(s)

Hank Stevens <HStevens@muohio.edu>

References


See Also

lvcomp3, clogistic, igp, scheffer

Examples

```r
## Specify the number of species
S <- 10
alpha <- .01
r <- runif(S)*2
a <- matrix(rnorm(S^2, m=alpha, sd=alpha/10), nrow=S, ncol=S)
parms <- list(r,a)
t=seq(0,40, by=.1)
N0 <- runif(S)/(S*alpha)
library(deSolve)
lvout <- ode(N0, t, lvcompg, parms)
matplot(t, lvout[,,-1], type="l", ylab="N", log="y")
```

LVCompGames

Lotka-Volterra Competition Games

Description

A pedagogical tool for learning about isoclines and stable equilibria.
Usage

LVCompGames(Alpha = NULL, r1 = 0.1, r2 = 0.1, num = 20, time = 10, step = 1)

Arguments

Alpha a 2 x 2 matrix of coefficients; if NULL, then a sensible random matrix will be generated - fun for games.

r1 the intrinsic rate of increase for species 1 (isocline – a solid line).

r2 the intrinsic rate of increase for species 2 (isocline – a dashed line).

num the desired number of random starting points for trajectories.

time the number of time steps for each trajectory.

step the number of time steps for which the integration is estimated (has no effect on accuracy, merely the smoothness of the line).

Value

First generates a plot of isoclines and initial points; the user is then prompted to "Hit <return>" in the console to see the trajectories. These will indicate the rate and path the trajectories, demonstrating, among other things, whether the equilibrium is stable.

Author(s)

Hank Stevens <HStevens@muohio.edu>

References


See Also

lvcomp2, lvcompg, clogistic,

Examples

## LVCompGames() # Hit return in the console to see the trajectories.
MetaSim  

Simulation of Stochastic Metapopulation Models

Description

Originally focused on creating a community of core-satellite species, this function allows simulation of several metapopulation models, where colonization and extinction rates are stochastic draws from uniform distributions, with specified means and ranges.

Usage

```r
MetaSim(
  Time = 50,
  NSims = 1,
  method = "hanski",
  ci = 0.25,
  e = 0.25,
  phi = 0.75,
  p0 = 0.5,
  D = 0.5
)
```

Arguments

- `Time`: A scalar for the number of time steps over which to simulate each population.
- `NSims`: A scalar for the number of simulations, which is analogous to the number of species in the community.
- `method`: A character string, in quotes, specifying which metapopulation model to use: "hanski", "gotelli", "lande", "levins". See "See Also" below.
- `ci`: Scalar for mean colonization rate.
- `e`: Scalar for mean extinction rate.
- `phi`: A scalar for the relative variability in rates. See Details.
- `p0`: Initial proportion of sites occupied for each species.
- `D`: Parameter for habitat destruction; applies to only the "lande" model.

Details

`phi` is one half of the relative range of each rate (colonization and extinction). For each time step, each rate is drawn from a uniform distribution, `Unif(rate-phi, rate+phi)`. Thus, the range is 2*phi, and center on the specified mean (`ci` or `e`).
Value

Function returns a list with these components.

- **method**: The method used (default is "hanski").
- **time**: The integer sequence of times, from 0 to the value of the argument Time.
- **Ns**: Time by NSims matrix of observed population sizes.
- **Parameters**: A named vector of the parameters used for the simulations.

Author(s)

Hank Stevens <HStevens@muohio.edu>

References


See Also

- gotelli
- hanski
- levins

Examples

```r
out <- MetaSim(NSims=2)
pops <- out$Ns
matplot(out$t, pops, type='l')
title(sub=paste(out$method, "model"))
```

---

**moths**

*Moth Species Richness*

Description

Data set of moth species richness from Ohio and Indiana, USA.

Format

A data frame with 21 observations on the following 6 variables.

- **region**: a factor with levels NCT WAP for the two collection regions.
- **site**: a factor with levels associated with county names.
- **area**: a numeric vector for the area of each forest fragment.
- **spp**: a numeric vector, the number of moth species.
- **lat**: a numeric vector; latitude
- **long**: a numeric vector; longitude
Source

Data provided graciously by Keith Summerville and Thomas Crist; see "A Primer of Ecology with R" for specific references.

References


Examples

data(moths)
plot(spp ~ area, data=moths)

---

**one_over_f**

*Function to generate 1/f noise.*

Description

Generates 1/f noise with a specified power or amplitude.

Usage

one_over_f(gamma = 1, N = 200)

Arguments

- **gamma**: spectral power, numeric, where 0 generates a white noise time series, 2 generates reddened noise. Defaults to 1 (pink).
- **N**: length of the time series.

Author(s)

Hank Stevens

References


See Also

[spec_mimic()] to rearrange one vector, X, to mimic the spectrum of another vector, Y; [plot_f()] to plot the time series and the spectrogram of the series.

Examples

```r
set.seed(1)
time.series <- one_over_f(gamma=2, N=50)
plot(1:50, time.series, type='l', main="Reddened noise")
time.series <- one_over_f(gamma=0, N=50)
plot(1:50, time.series, type='l', main="White noise")
```

---

**Description**

Used primarily to repeat simulations and analyses of Pimm and Lawton (1977), given a Jacobian matrix. Analyses include eigenanalysis, but also measuring average interaction strength (May 1972), average intraspecific negative density dependence, and the strength of the omnivory interaction, if present.

**Usage**

```r
pimmlawton(mat, N = 1, omni.i = NA, omni.j = NA, omega = NULL)
```

**Arguments**

- `mat`: a numerical matrix; the "maximum" Jacobian matrix. See details below.
- `N`: a scalar for the number of randomizations
- `omni.i`: if omnivory is present, the row/col index for the prey.
- `omni.j`: if omnivory is present, the row/col index for the predator.
- `omega`: if not NULL, a scalar $0<x<1$ indicating the relative weight of omnivory (after McCann et al. (1998))

**Details**

This function simulates a constrained randomization of a Jacobian food web matrix. The matrix it uses `mat` is of a special form, which assumes that all non-zero values are drawn from a uniform distribution between zero and a value of some specified magnitude, either positive or negative.
Value

Returns a data frame, where each row corresponds to a single random Jacobian matrix, with the following columns.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DomEig</td>
<td>the real part of the dominant eigenvalue</td>
</tr>
<tr>
<td>Im</td>
<td>the imaginary part of the dominant eigenvalue</td>
</tr>
<tr>
<td>IntraDD</td>
<td>average magnitude, over all species, of the intraspecific negative density dependence; the square root of the sum of the squared diagonal elements of the random Jacobian matrix</td>
</tr>
<tr>
<td>I</td>
<td>average interaction strength (after May 1972); the square root of, sum of the squared off diagonal elements divided by the number of off diagonal elements.</td>
</tr>
<tr>
<td>I.omni</td>
<td>average interaction strength for the omnivory interaction; the square root of, sum of the squared omnivory elements effect of predator on prey and prey on predator divided by two.</td>
</tr>
</tbody>
</table>

Author(s)

Hank Stevens <HStevens@muohio.edu>

References


Examples

### A relevant style of matrix for the first food chain in Pimm and Lawton (1977).
### Note each non-zero element is the appropriate sign, and the maximum magnitude specified by Pimm and Lawton (1977).

```r
Aq = matrix(c(-1, -10, 0, 0,
             0.1, 0, -10, 0,
             0, 0.1, 0, -10,
             0, 0, 0.1, 0),
            nrow=4, byrow=TRUE)
pimmlawton(Aq, N=1)
```

```r
out <- pimmlawton(Aq, N=2000)
out <- subset(out, -1/DomEig <150)
hist(-1/out$DomEig, main="Frequencies of Return Time")
```
plot_f

Description

Function to plot a time series and its periodogram, and calculate the confidence interval of 2 times the spectral power.

Usage

plot_f(z)

Arguments

z numeric vector.

Author(s)

Hank Stevens

References


See Also

[one_over_f()]{hl} to generate 1/f noise; [spec_mimic()] to rearrange one vector, X, to mimic the spectrum of another vector; [spectrum()] for the hard work.

Examples

```r
## white noise
plot_f(z=runif(50))
```
PopSim  Population Simulator

Description

Resampling stochastic simulator for a single density-independent population.

Usage

PopSim(Rs, N0, years = 50, sims = 10)

Arguments

Rs  vector of observed annual growth rates (N[t+1]/N[t]).
N0  initial population size.
years  number of years to simulate.
sims  number of simulated populations.

Details

Designed to simulate trajectories based on resampled observed N[t+1]/N[t].

Value

Returns a matrix of population sizes for time = t (rows) for each replicated population (columns).

References


See Also

sparrows

Examples

```r
lambdas <- runif(10, .5, 1.5)
out <- PopSim(Rs=lambdas, years=50, N0=50)
matplot(0:50, out, type='l')
summary(out[51,])
```
The Lotka-Volterra Predator-prey Model

Description

The Lotka-Volterra predator-prey model, for use with ode in the deSolve package.

Usage

predpreyLV(t, y, params)

Arguments

- t: Argument for time
- y: A vector of length 2, for population 1 and 2
- params: Vector or list of parameters

Value

Returns a list of length one, for use with ode in the deSolve package.

Author(s)

Hank Stevens <HStevens@muohio.edu>

References


See Also

predpreyRM

Examples

```r
params1 <- c(b=.5, a=.01, s=.2, e=.1)
Time <- seq(0,100, by=.1) # Set time here
LV.out <- ode(c(H0=25,P0=5), Time, predpreyLV, params1)
matplot(Time, (LV.out[,2:3]), type="l", ylab="Population Size")
```
Rosenzweig-MacArthur Predator-prey Model

Description

An implementation of a predator-prey model, after Rosenzweig and MacArthur (1963). Includes prey density-dependence, and a type-II predator functional response. For use with ode in the deSolve package.

Usage

predpreyRM(t, y, p)

Arguments

t       Argument for time
y       A vector of length 2, for population 1 and 2
p       Vector or list of parameters

Value

Returns a list of length one, for use with ode in the deSolve package.

Component 1  vector of the state variables, y.

Author(s)

Hank Stevens <HStevens@muohio.edu>

References


See Also

predpreyLV

Examples

library(deSolve)
pars <- c(b = .8, e = 0.07, s = .2, w = 5, D = 400, alpha = 0.001)
Time <- 50
RM1 <- ode(c(900,120), 1:Time, predpreyRM, pars)
matplot(1:Time, RM1[-1], type='l')
Description

Kermack and McCormick (1927) provided data on the number of plague deaths per week in Bombay in 1905–06 (Bombay is the former name for the Indian coastal city Mumbai. It is the capital of Maharashtra and is one of the largest cities in the world).

Format

A data frame with 32 observations on the following 2 variables.

- **Week** a numeric vector
- **CumulativeDeaths** a numeric vector

Source

Data provided kindly by S.P. Ellner (Cornell University)

References


Examples

```r
data(ross)
str(ross) ; plot(CumulativeDeaths ~ Week, data=ross, type='b')
```

Description

A model of floating vs. submerged plant dominance in shallow aquatic systems, after Scheffer et al. (2003). For use with ode in the deSolve package. Floating plants are better competitors for light, as long as submerged plants cannot drive down nitrogen levels.

Usage

```r
scheffer(t, y, p)
```
Arguments

- `t` the time point for a realization of the integration.
- `y` the vector of populations, at each time `t`.
- `p` a vector or list containing the necessary parameters.

Value

Returns a list of length one which is the vector of the rates of increase (required by `ode`).

Author(s)

Hank Stevens <HStevens@muohio.edu>

References


See Also

- `lvcompg`, `igp`

Examples

```r
p <- c(N=2.5, as=0.01, af=0.01, b=0.02, qs=0.075, qf=0.005,
       hs=0, hf=0.2, ls=0.05, lf=0.05, rs=0.5, rf=0.5, W=0)
t <- 1:200
Initial <- c(F=10, S=10)
S.out1 <- ode(Initial, t, scheffer, p)
matplot(t, S.out1[-1], type='l')
legend('right', c("F", "S"), lty=1:2, col=1:2, bty='n')
```

SIRbd is the S-I-R Epidemiological Disease Model with Births and Deaths

Description

The S-I-R epidemiological disease model with births and deaths (population dynamics), for use with `ode` in the deSolve package. This model uses mass action transmission.

Usage

SIRbd(t, y, p)
**Arguments**

- `t`: times points for which values will be returned
- `y`: the vector of disease states of hosts (S, I, R)
- `p`: a vector of parameters

**Details**

The user does not put these directly into this function, but rather uses `ode` in the `deSolve` package.

**Value**

Returns of list of one component (required by `ode`).

**Author(s)**

Hank Stevens <HStevens@muohio.edu>

**References**


**See Also**

`ross`, `SIRf`, `SIRd`

**Examples**

```r
library(deSolve)
N <- 10^6; R <- 0; I <- 1; S <- N - I - R
g <- 1/(13/365); b <- 1/50;
age <- 5; R0 <- 1 + 1/(b*age)
B <- R0 * (g + b) / N
parms <- c(B = B, g = g, b = b, m=b)
years <- seq(0,30, by=.1)
SIRbd.out <- data.frame(ode(c(S=S,I=I,R=R), years, SIRbd, parms, hmax=.01))
matplot(SIRbd.out[,1], sqrt(SIRbd.out[,2:3]), type='l',
lty=1:3, ylab="sqrt(No. of Individuals)", xlab='Years')
legend('right', c('S', 'I', 'R'), lty=1:3, col=1:3, bty='n')
```
The S-I-R Epidemilogical Disease Model

Description
The S-I-R epidemiological disease model with density-dependent transmission, for use with ode in the deSolve package.

Usage
SIRd(t, y, p)

Arguments
- t: times points for which values will be returned
- y: the vector of disease states of hosts (S, I, R)
- p: a vector of parameters

Details
The user does not put these directly into this function, but rather uses ode in the deSolve package.

Value
Returns of list of one component (required by ode).

Author(s)
Hank Stevens <Hank.Stevens@miamioh.edu>

References

See Also
ross, SIRf, SIRbd
Examples

N <- 10^3; I <- R <- 1; S <- N - I - R
parms <- c(B=.01, g=4)
months <- seq(0, 3, by=0.01)
require(deSolve)
SIR.out <- data.frame( ode(c(S,I,R), months, SIRD, parms) )
matplot(months, SIR.out[,,-1], type='l'

legend('right', c('R', 'I', 'S'), lty=3:1, col=3:1, bty='n')

SIRf

The S-I-R Epidemiological Disease Model with Frequency Dependent Transmission

Description

The S-I-R epidemiological disease model with frequency dependent transmission, for use with ode in the deSolve package.

Usage

SIRf(t, y, p)

Arguments

t  times points for which values will be returned
y  the vector of disease states of hosts (S, I, R)
p  a vector of parameters

Details

The user does not put these directly into this function, but rather uses ode in the deSolve package.

Value

Returns of list of one component (required by ode).

Author(s)

Hank Stevens <HStevens@muohio.edu>

References

SIRmod

See Also

ross, SIRD, SIRbd, ode

Examples

```r
R <- 0; S <- 1000; I <- 1000; N <- S+I+R
parmsf <- c(B=1, g=1)
Months <- seq(0, 8, by=0.1)
outf <- ode(c(S,I,R), Months, SIRf, parmsf)
matplot(Months, outf[,,-1], type='l', ylab="Prevalence (I/N)")
legend('right', legend=c('S', 'I', 'R'), lty=1:3, col=1:3, bty='n')
```

SIRmod

The S-I-R Epidemiological Disease Model

Description

The S-I-R epidemiological disease model with births and deaths (population dynamics), for use with ode in the deSolve package. This model uses scaled transmission, where $z$ controls the degree of density- and frequency-dependence.

Usage

```r
SIRmod(t, y, p)
```

Arguments

- `t`: times points for which values will be returned
- `y`: the vector of disease states of hosts (S, I, R)
- `p`: a vector of parameters

Details

The user does not put these directly into this function, but rather uses ode in the deSolve package.

Value

Returns of list of one component (required by ode).

Author(s)

Hank Stevens <Hank.Stevens@miamioh.edu>
sparrows

References

Kermack, W.O. and McCormick, W.G. (1927) A contribution to the mathematical theory of epidem-

See Also

ross, SIRf, SIRd

Examples

library(deSolve)
N <- 10^6; R <- 0; I <- 1; S <- N - I - R
g <- 1/(13/365); b <- 1/50; z <- 0;
age <- 5; R0 <- 1 + 1/(b*age)
B <- R0 * (g + b) / N
parms <- c(B = B, g = g, b = b, mu=b)
years <- seq(0,30, by=.1)
SIR.out <- data.frame(ode(c(S=S,I=I,R=R), years, SIRmod, parms, hmax=.01))
matplot(SIR.out[,1], sqrt(SIR.out[-1]), type='l',
        lty=1:3, ylab="sqrt(No. of Individuals)", xlab='Years')
legend('right', c('S','I','R'), lty=1:3, col=1:3, bty='n')

sparrows  Song Sparrow Data Set

Description

Song Sparrow (Melospiza melodia) counts in Darlington, OH, USA. From Sauer, J. R., J.E. Hines,
Version 2005.2. USGS Patuxent Wildlife Research Center, Laurel, MD.

Format

A data frame with 36 observations on the following 3 variables.

Year  a numeric vector
Count  a numeric vector
ObserverNumber  a numeric vector

Source

http://www.pwrc.usgs.gov/BBS/
References


See Also

PopSim

Examples

```r
data(sparrows)
## maybe
str(sparrows)
plot(Count ~ Year, sparrows)
```

---

**spec_mimic**

*Function to mimic the power spectrum of an observed time series.*

Description

This function rearranges one vector, `X`, to mimic the spectrum of another vector, `Y`.

Usage

```r
spec_mimic(X, Y = NULL, gamma = 1)
```

Arguments

- `X`: numeric vector.
- `Y`: if not NULL, a numeric vector.
- `gamma`: power of the 1/f noise; used only if `Y` is NULL.

Details

If `Y` is NULL, this function will use `one_over_f()` to generate a random series with power = `gamma`.

Author(s)

Hank Stevens

References

See Also

[one_over_f()] to generate 1/f noise; [plot_f()] to plot the time series and the spectrogram of the series.

Examples

N = 50
set.seed(1)
X1 <- runif(N)
Y <- one_over_f(gamma=2, N=N)
X2 <- spec_mimic(X1, Y)
series <- cbind(X1, Y, X2)
matplot(1:50, series, type='l')
legend("bottomright", legend=colnames(series), col=1:3, lty=1:3)

succniche

A Four-state model of Successional Dynamics

Description

This is the five-state, two-species model of the succession-niche model, after Pacala and Rees (1998). For use with ode in the deSolve package.

Usage

succniche(t, y, params)

Arguments

t Argument for the time point at integration
y A vector of length four, for states, E, M, S, and R.
params Vector or list of parameters

Value

Returns a list of length one, for use with ode in the deSolve package.

Component 1 vector of the state variable (a scalar for the proportion of sites occupied).

Author(s)

Hank Stevens <HStevens@muohio.edu>

References

thetalogistic

See Also
levins, compcol, compcolM

Examples

```r
params.suc <- c(a=7, c=0.2, g=.1, m=0.04, D=0)
t=seq(0,50,.1)
init.suc <- c(S=0, E=0.5, M=0.5, R=0.00) # Free space is implicit, F=1-(S+E+M+R).
ccg.out <- data.frame(ode(init.suc, t, succniche, params.suc))
matplot(t, ccg.out[,,-1], type="l", ylab="Relative Frequency",
xlab="Time", ylim=c(0,1) )
legend("right", colnames(ccg.out)[5:2], lty=4:1, bty="n")
```

---

thetalogistic  

Continuous Theta-Logistic Growth

Description

A function for continuous theta-logistic growth, for use with ode in the deSolve package.

Usage

```r
thetalogistic(times, y, parms)
```

Arguments

- `times`  
  Times points that will return N
- `y`  
  N
- `parms`  
  a vector of logistic growth parameters

Details

The user does not put these directly into this function, but rather uses ode in the deSolve package. See ode in the deSolve package.

Value

Returns of list of one component (required by ode).

Author(s)

Hank Stevens (HStevens@muohio.edu)
weeds

References


See Also
clogistic

Examples

```r
library(deSolve)
p <- c(r=1, alpha=.01, theta=.5)
time <- seq(1, 10, by=.1)
initialN <- 10
out <- ode(y=initialN, times=time, func=thetalogistic, parms=p)
plot(time, out[, -1], type='l')
```

---

weeds

Percent cover of six perennial herbaceous plants

Description

Percent cover of six of the most common herbaceous perennial species from the Buell-Small long term succession study (http://www.ecostudies.org/bss/). Data are plot-level visual estimates of percent cover.

Format

A data frame with 15140 observations on the following 8 variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>a numeric vector</td>
</tr>
<tr>
<td>Genus</td>
<td>a factor with levels Aster, Euthamia, Solidago</td>
</tr>
<tr>
<td>Epithet</td>
<td>a factor with levels canadensis, gigantea, graminifolia, novae-angliae, pilosus, rugosa</td>
</tr>
<tr>
<td>FieldName</td>
<td>a factor with levels C3, C4, C5, C6, C7, D1, D2, D3, E1, E2</td>
</tr>
<tr>
<td>Age</td>
<td>a numeric vector indicating the number of years of succession, since abandonment from agriculture.</td>
</tr>
<tr>
<td>PlotId</td>
<td>a numeric vector</td>
</tr>
<tr>
<td>Cover</td>
<td>a numeric vector; percent cover, estimated visually.</td>
</tr>
<tr>
<td>Species</td>
<td>a factor with levels A.novae-angliae, A.pilosus, E.gigantea, S.canadensis, S.gigantea, S.rugosa</td>
</tr>
</tbody>
</table>
Source

Data graciously provided by Scott Meiners (Eastern Illinois University, http://www.ecostudies.org/bss/).

References


Examples

data(weeds)
str(weeds);
# lattice::xyplot(Cover ~ Age, data=weeds, groups=Species,
# type=c("a"), auto.key=list(lines=TRUE, points=FALSE, columns=3),
# ylim=c(-1,20))
Index

* 1/f
  one_over_f, 30
  plot_f, 33
  spec_mimic, 44
* Allee
  alogistic, 3
* cats
  cat_function, 7
* color-mimicry
  plot_f, 33
* color
  one_over_f, 30
  spec_mimic, 44
* datasets
  BSsucc, 6
  CandG, 7
  ClostExp, 11
  coneflower, 13
  coneflowerrecruits, 14
  coneflowerseeds, 15
  moths, 29
  ross, 37
  sparrows, 43
  weeds, 47
* effect
  alogistic, 3
* logistic
  alogistic, 3
* methods
  bip_stability, 4
  chesson, 8
  clogistic, 9
  compcol, 11
  compcolM, 12
  dlogistic, 15
  dlvcomp2, 16
  gotelli, 17
  hanski, 18
  igp, 19
  lande, 20
  levins, 22
  lvcomp2, 23
  lvcomp3, 24
  lvcompG, 25
  LVCompGames, 26
  MetaSim, 28
  pimmlawton, 31
  PopSim, 34
  predpreyLV, 35
  predpreyRM, 36
  scheffer, 37
  SIRbd, 38
  SIRd, 40
  SIRf, 41
  SIRmod, 42
  succniche, 45
  thetalogistic, 46
* mimicry
  spec_mimic, 44
* noise
  one_over_f, 30
  plot_f, 33
  spec_mimic, 44
* package
  primer-package, 2
* periodogram
  plot_f, 33
* spectra
  one_over_f, 30
  plot_f, 33
  spec_mimic, 44
  alogistic, 3
  bip_stability, 4
  BSsucc, 6
  CandG, 7
  cat_function, 7
chesson, 8
clogistic, 4, 9, 16, 18, 19, 21–23, 25–27, 47
ClostExp, 11
compcol, 11, 13, 46
compcolM, 12, 12, 46
coneflower, 13
coneflowerrecruits, 14
coneflowerseeds, 15
dlogistic, 4, 10, 15, 17
dlvcomp2, 16
gotelli, 17, 18, 19, 21, 22, 29
hanski, 18, 18, 19, 21, 22, 29
igp, 19, 26, 38
lande, 18, 19, 20, 22
levins, 4, 10, 12, 13, 21, 22, 29, 46
dlvcmp2, 10, 17, 23, 25, 27
dlvcmp3, 17, 23, 24, 26
dlvcmpg, 3, 10, 16, 17, 20, 23, 25, 27, 38
LVCompGames, 26
MetaSim, 18, 19, 21, 22, 28
moths, 29
ode, 3, 42
one_over_f, 30
pimmlawton, 31
plot_f, 33
PopSim, 34, 44
predpreyLV, 35, 36
predpreyRM, 35, 36
primer (primer-package), 2
primer-package, 2
ross, 37, 39, 40, 42, 43
scheffer, 20, 26, 37
SIRbd, 38, 40, 42
SIRd, 39, 40, 42, 43
SIRf, 39, 40, 41, 43
SIRmod, 42
sparrows, 34, 43
spec_mimic, 44
succniche, 9, 12, 13, 45
thetalogistic, 4, 10, 46
weeds, 47