Package ‘wiqid’

June 10, 2020

Type Package

Title Quick and Dirty Estimates for Wildlife Populations

Version 0.3.0

Date 2020-06-10

Depends HDInterval, R (>= 2.10), mcmcOutput

Imports stats, truncnorm, MASS, coda, plotrix

Suggests secr (>= 3.0), shiny, rjags

BugReports https://github.com/mikemeredith/wiqid/issues

URL https://mmeredith.net/R/wiqid/

Description Provides simple, fast functions for maximum likelihood and Bayesian estimates of wildlife population parameters, suitable for use with simulated data or bootstraps. Early versions were indeed quick and dirty, but optional error-checking routines and meaningful error messages have been added. Includes single and multi-season occupancy, closed capture population estimation, survival, species richness and distance measures.

Language en-GB

License GPL-3

NeedsCompilation no

Author Mike Meredith [aut, cre],
  Jason Bryer [ctb] (showShinyApp),
  John Kruschke [ctb],
  Brian Neelon [ctb] (Bnormal),
  Michael Schaub [ctb] (ch2mArray),
  R Core Team [ctb] (stats::AIC code adapted for AICc)

Maintainer Mike Meredith <mike@mmeredith.net>

Repository CRAN

Date/Publication 2020-06-10 11:00:03 UTC
R topics documented:

wiqid-package ................................................. 3
AICc .......................................................... 7
AICtable .......................................................... 9
allCombinations ................................................. 10
Bayesian binomial analysis ................................. 11
Bayesian normal estimation ................................ 12
Bayesian Occupancy Single Season .......................... 14
Bayesian Poisson analysis ..................................... 17
Bayesian SCR .................................................... 18
BetaDist .......................................................... 19
Bwiqid-class ...................................................... 22
Closed Captures .................................................. 22
deprecated ......................................................... 25
dippers ............................................................. 26
Distance Measures ............................................... 27
distShell .......................................................... 30
distTestData ......................................................... 31
Diversity indices .................................................. 32
GammaDist .......................................................... 34
getMCerror ......................................................... 35
GrandSkinks ......................................................... 36
KanhaTigers ......................................................... 37
KillarneyBirds ....................................................... 38
Links ................................................................. 39
MeadowVoles ....................................................... 40
occ2sps ............................................................ 41
Occupancy Multi-Season ......................................... 42
Occupancy Single Season ......................................... 44
plot.Bwiqid ......................................................... 47
plotACs ............................................................. 48
plotComb ............................................................ 49
predict.wiqid ....................................................... 51
print.Bwiqid ......................................................... 53
Priors ................................................................. 54
railSims ............................................................. 55
richCurve ............................................................ 56
richRarefy ........................................................... 58
Royle-Nichols occupancy model .............................. 59
salamanders ......................................................... 61
secrFit ............................................................... 61
seedbank ............................................................. 62
showShinyApp ....................................................... 63
simpleRhat ........................................................ 64
Species richness estimators ..................................... 65
standardize ........................................................ 67
Survival (CJS) ....................................................... 69
Fast, simple estimation functions for wildlife population models

Description

Quick and dirty functions to estimate occupancy, survival, abundance, species richness and diversity, etc. for wildlife populations.

Details

There are a number of sophisticated programs for the analysis of wildlife data, producing estimates of occupancy, survival, abundance, or density. wiqid began as a collection of fast, bare-bones functions which can be run from R suitable for use when you are generating hundreds of simulated data sets. The package takes its name from the quick-and-dirty nature of the original functions.

We now use wiqid in basic wildlife study design and data analysis workshops, and most functions now have options to check the input data and give informative error messages. Workshop participants have used lm, glm and functions in the secr and BEST packages. So wiqid tries to match the look and feel of these functions.

All functions use standard data frames or matrices for data input. ML estimation functions return objects of class wiqid with parameter estimates on the transformed scale (usually logit functions), variance-covariance matrix, and back-transformed ‘real’ values; there are print, logLik and predict methods. Bayesian functions (distinguished by an initial “B”) return class mcmcOutput objects.

Simulations and bootstraps often generate weird data sets, eg. capture histories with no captures. These functions do not throw errors or give warnings if the data are weird, but return NAs if estimates cannot be calculated. Errors may still occur if the data are impossible, eg. 6 detections in 5 occasions.

Note that in version 0.2.0 the scaling of continuous covariates has changed to SD=1 (previously SD=0.5). This means that beta coefficients will now be exactly half the size, matching the output from other software.

The functions are listed by topic below.

SIMPLE BAYESIAN POSTERIORS

- Bbinomial: generate draws from a conjugate beta posterior distribution
- Bpoisson: generate draws from a conjugate gamma posterior distribution
- Bnormal: fit a basic normal model to data
OCCUPANCY

Single-season occupancy

- **occSS**: general-purpose ML function; allows site- and survey-specific covariates
- **BoccSS**: general-purpose Bayesian implementation of the above
- **occSS0**: a basic psi(.) p(.) model, faster if this is all you need
- **BoccSS0**: a Bayesian implementation of the psi(.) p(.) model
- **occSSrn**: Royle-Nichols method
- **occSStime**: faster if you have only time effects, also does a plot
- **occSScovSite**: faster if you only have site-specific covariates
- **occ2sps**: single-season two-species models

Multi-season occupancy

- **occMS**: general-purpose function; parameters depend on covariates; slow
- **occMScovSite**: smaller range of covariate options
- **occMS0**: a simple multi-season model with four parameters; faster
- **occMStime**: parameters vary by season; faster

DENSITY from spatial capture-recapture data

We use the [secr](https://ssecr.github.io/) package for ML estimation of density. For Bayesian estimation, **wiqid** offers:

- **Bsecr0**: a Bayesian implementation of the intercept-only model

ABUNDANCE from closed-population capture-recapture data

Although data for genuinely closed populations are rare, this is an important conceptual stepping-stone from CJS models to robust models for survival.

- **closedCapM0**: simple model with constant capture probability
- **closedCapMb**: permanent behavioural response to first capture
- **closedCapMt**: capture probability varies with time
- **closedCapMtcov**: allows for time-varying covariates
- **closedCapMh2**: heterogeneity with 2-mixture model
- **closedCapMhJK**: jackknife estimator for heterogeneity

SURVIVAL from capture-recapture data

Cormack-Jolly-Seber models

- **survCJS**: model with time-varying covariates
- **BsurvCJS**: a Bayesian implementation of the above
- **survCJSaj**: allows for different survival for adults and juveniles

Pollock’s robust design
survRDah 2-stage estimation of survival and recruitment
survRD  single stage maximum likelihood estimation

Note that the RD functions are preliminary attempts at coding these models and have not been fully
tested or benchmarked.

SPECIES RICHNESS from species x sample matrices

Rarefaction

richRarefy  Mao’s tau estimator for rarefaction
richCurve   a shell for plug-in estimators, for example...
richSobs    the number of species observed
richSingle  the number of singletons observed
richDouble  the number of doubletons observed
richUnique  the number of uniques observed
richDuplicate  the number of duplicates observed

Coverage estimators

richACE     Chao’s Abundance-based Coverage Estimator
richICE     Chao’s Incidence-based Coverage Estimator
richChao1   Chao1 estimator
richChao2   Chao2 estimator
richJack1   first-order jackknife estimator
richJack2   second-order jackknife estimator
richJackA1  abundance-based first-order jackknife estimator
richJackA2  abundance-based second-order jackknife estimator
richBoot    bootstrap estimator
richMM      Michaelis-Menten estimator
richRenLau  Rennolls and Laumonier’s estimator

BIODIVERSITY INDICES

Alpha diversity
All of these functions express diversity as the number of common species in the assemblage.

biodSimpson  inverse of Simpson’s index of dominance
biodShannon  exponential form of Shannon’s entropy
biodBerger   inverse of Berger and Parker’s index of dominance
biodBrillouin exponential form of Brillouin’s index

Beta diversity / distance
All of these functions produce distance measures (not similarity) on a scale of 0 to 1. The function
distShell provides a wrapper to produce a matrix of distance measures across a number of sites.

distBrayCurtis complement of Bray-Curtis index, aka ‘quantitative Sorensen’
distChaoJaccCorr complement of Chao’s Jaccard corrected index
distChaoJaccNaive complement of Chao’s Jaccard naive index
distChaoSorCorr complement of Chao’s Sorensen corrected index
distChaoSorNaive complement of Chao’s Sorensen naive index
distChord distance between points on a normalised sphere
distJaccard complement of Jaccard’s index of similarity
distMorisitaHorn complement of the Morisita-Horn index of similarity
distOchiai complement of the Ochiai coefficient of similarity
distPreston Preston’s coefficient of faunal dissimilarity
distRogersTanimoto complement of the Rogers and Tanimoto’s coefficient of similarity
distSimRatio complement of the similarity ratio
distSorensen complement of the Sorensen or Dice index of similarity
distWhittaker Whittaker’s index of association

DATA SETS

dippers Capture-recapture data for European dippers
distTestData artificial data set for distance measures
GrandSkinks multi-season occupancy data
KanhaTigers camera-trap data for tigers
KillarneyBirds abundance of birds in Irish woodlands
MeadowWoles mark-recapture data from a robust design study
railSims simulated detection/non-detection data for two species of rails
salamanders detection/non-detection data for salamanders
seedbank number of seeds germinating from samples of soil
Temburong counts of tree species in a 1ha plot in Brunei
TemburongBA basal area of tree species in a 1ha plot in Brunei
weta detection/non-detection data and covariates for weta

DISTRIBUTIONS

These are convenience wrappers for the related d/p/q/r functions in the stats package which allow for parameterisation with mean and sd or (for Beta) mode and concentration.

dbeta2 etc Beta distribution with mean and sd
dbeta3 etc Beta distribution with mode and concentration
dgamma2 etc Gamma distribution with mean and sd
dt2 etc t-distribution with location, scale and df parameters
dt3 etc t-distribution with mean, sd and df parameters

UTILITY FUNCTIONS

AICc AIC with small-sample correction
AICtable tabulate AIC for several models
allCombinations model formulae for combinations of covariates
standardize a simple alternative to scale.
AICc

Author(s)
Mike Meredith

---

**AICc**

*Akaike’s Information Criterion with small-sample correction - AICc*

---

**Description**

Akaike’s Information Criterion with small-sample correction - AICc

**Usage**

AICc(object, ..., nobs, df)

**Arguments**

- **object**: a fitted model object for which there exists a logLik method to extract the corresponding log-likelihood, number of parameters, and number of observations.
- **...**: optionally more fitted model objects.
- **nobs**: scalar; the value to use for the effective sample size; overrides the value contained in the model object(s).
- **df**: the value to use for the number of parameters; usually a vector of length = number of models; non-NA elements override the value contained in the corresponding model object.

**Details**

AICc is Akaike’s information Criterion (AIC) with a small sample correction. It is

\[
AIC_{c} = AIC + 2K(K + 1)/(n - K - 1)
\]

where \(K\) is the number of parameters and \(n\) is the number of observations.

This is an S3 generic, with a default method which calls logLik, and should work with any class that has a logLik method.

**Value**

If just one object is provided, the corresponding AICc.

If multiple objects are provided, a data frame with rows corresponding to the objects and columns representing the number of parameters in the model (df) and the AICc.

The result will be Inf for over-parameterised models, i.e. when df >= nobs -1.
Note

For some data types, including occupancy data, there is debate on the appropriate effective sample size to use.

Author(s)

Essentially the same as AIC in package stats. Modified to return AICc by Mike Meredith.

References


See Also

AIC.

Examples

# Occupancy models
data(salamanders)
mt <- occSStime(salamanders, p ~ .time, plot=FALSE)
MT <- occSStime(salamanders, p ~ .Time, plot=FALSE)
AIC(mt, MT)
AICc(mt, MT)
# The default sample size = the number of sites
nobs(mt) == nrow(salamanders)
# It is sometimes taken to be the total number of surveys...
AICc(mt, MT, nobs=length(salamanders))
# ... or the minimum of...
n <- min(sum(rowSums(salamanders) > 0), # sites where species was detected
       sum(rowSums(salamanders) == 0)) # sites where species was not detected
AICc(mt, MT, nobs=n)

# Survival models
data(dippers)
DH <- dippers[1:7] # Extract the detection histories
null <- survCJS(DH) # the phi(.) p(.) model
phit <- survCJS(DH, phi ~ .time) # the phi(t) p(.) model
full <- survCJS(DH, list(phi ~ .time, p ~ .time)) # the phi(t) p(t) model
AICc(null, phit, full)
# for the full model, all 12 parameters cannot be estimated;
# we can manually set df=11 for this model:
AICc(null, phit, full, df=c(NA, NA, 11))
**AICtable**

**Make a table for AIC or other criterion**

**Description**
Takes the output from a call to AIC or AICc returns a data frame with model likelihoods and model weights.

**Usage**

\[ \text{AICtable}(x, \text{digits}=3, \text{sort}) \]

**Arguments**

- **x**: A data frame with the second column being AIC-type values, as returned by AIC or AICc.
- **digits**: integer indicating the number of decimal places to retain.
- **sort**: logical indicating whether the table should be sorted by AIC; if missing, the rows are sorted if the table has row names.

**Value**

Returns the original data frame with the following extra columns

- **Delta**: The difference between each value in \( x \) and \( \text{min}(x) \)
- **ModelLik**: The model likelihood
- **ModelWt**: The model weight

If sort = TRUE, the rows will be sorted by increasing values of AIC/AICc.

**Author(s)**

Mike Meredith

**References**


**Examples**

```r
data(KanhaTigers)
mods <- NULL
mods$M0 <- closedCapM0(KanhaTigers)
mods$Mh2 <- closedCapMh2(KanhaTigers)
mods$MhJK <- closedCapMhJK(KanhaTigers)
mods$Mt <- closedCapMt(KanhaTigers)
AICc <- sapply(mods, AICc)
```
allCombinations

Create formulae for all combinations of covariates

**Description**

Create formulae for all combinations of covariates, currently main effects only.

**Usage**

```r
allCombinations(response = "", covars, formulae = TRUE)
```

**Arguments**

- `response`: a character vector of length 1 specifying the response variable.
- `covars`: a character vector specifying the covariates/predictors.
- `formulae`: if TRUE, only the formulae are returned; otherwise a TRUE/FALSE matrix is returned, with the formulae as row names.

**Value**

If `formulae = TRUE`, returns a character vector with elements corresponding to the formulae for all possible combinations of main effects.

Otherwise, returns a TRUE/FALSE matrix indicating which covariates are included in each model with the model formulae as the row names.

**Author(s)**

Mike Meredith

**Examples**

```r
longNames <- colnames(swiss)
# these would produce formulae too long for the console.
names(swiss) <- abbreviate(longNames)
vars <- colnames(swiss)
vars

# Get the formulae for all combinations of covars:
formulae <- allCombinations(vars[1], vars[-1])
formulae[1:10]

# Run all the models with 'lm', put results into a list:
# lms <- lapply(formulae, lm, data=swiss) # This works, but the call is a mess!
lms <- vector('list', 32)
for(i in 1:32)
```

```r
```
Bayesian binomial analysis

Bayesian analysis of binomial data

Description

Draws random values from the posterior for a binomial likelihood and beta prior. Bbinom is deprecated and will be removed in the next version; use Bbinomial instead.

Usage

Bbinom(...)
Bbinomial(y, n, priors=NULL, draws=100000, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>the number of successes</td>
</tr>
<tr>
<td>n</td>
<td>the number of trials</td>
</tr>
<tr>
<td>priors</td>
<td>an optional list with elements specifying the priors for the mode and concentration of the beta prior distribution; see Details.</td>
</tr>
<tr>
<td>draws</td>
<td>the number of MCMC draws to be returned.</td>
</tr>
<tr>
<td>...</td>
<td>other arguments to pass to the function.</td>
</tr>
</tbody>
</table>
Details

The function generates a vector of random draws from the posterior distribution of the probability of success. It uses conjugacy to determine the parameters of the posterior beta distribution, and draws independent values from this.

A prior can be specified with the priors argument. A beta prior is used, specified by mode, mode, and concentration, conc.

When priors = NULL (the default), a uniform prior corresponding to beta(1, 1) is used.

Value

Returns an object of class mcmcOutput.

Author(s)

Mike Meredith.

Examples

# Generate a sample from a binomial distribution, maybe the number of sites
# where a species was detected:
# number of trials (sites visited)
(n <- 10) # number of successes (sites where species detected)
(y <- rbinom(1, n, 0.75))
Bbinomial(y, n) # with uniform prior
Bbinomial(y, n, priors=list(mode=0.4, conc=5)) # with informative prior

Bayesian normal estimation

Bayesian modelling of a normal (Gaussian) distribution

Description

Bayesian estimation of centre ($\mu$) and scale (spread) ($\sigma$) of a normal distribution based on a sample. Bnormal uses a gamma prior on the precision, $\tau = 1/\sigma^2$, while Bnormal2 applies a gamma prior to $\sigma$.

Usage

Bnormal(y, priors=NULL,
    chains=3, draws=10000, burnin=100, ...)

Bnormal2(y, priors=NULL,
    chains=3, draws=3e4, burnin=0, thin=1, adapt=1000,
    doPriorsOnly=FALSE, parallel=NULL, seed=NULL, ...)
Bayesian normal estimation

Arguments

- **y**
  a vector (length > 1) with observed sample values; missing values not allowed.

- **priors**
  an optional list with elements specifying the priors for the centre and scale; see Details.

- **doPriorsOnly**
  if TRUE, Bnormal2 returns MCMC chains representing the prior distributions, *not* the posterior distributions for your data set.

- **chains**
  the number of MCMC chains to run.

- **draws**
  the number of MCMC draws per chain to be returned.

- **thin**
  thinning rate. If set to n > 1, n steps of the MCMC chain are calculated for each one returned. This is useful if autocorrelation is high.

- **burnin**
  number of steps to discard as burn-in at the beginning of each chain.

- **adapt**
  number of steps for adaptation.

- **seed**
  a positive integer (or NULL): the seed for the random number generator, used to obtain reproducible output if required.

- **parallel**
  if NULL or TRUE and > 3 cores are available, the MCMC chains are run in parallel. (If TRUE and < 4 cores are available, a warning is given.)

- **...**
  other arguments to pass to the function.

Details

The function generates vectors of random draws from the posterior distributions of the population centre (μ) and scale (σ). Bnormal uses a Gibbs sampler implemented in R, while Bnormal2 uses JAGS (Plummer 2003).

Priors for all parameters can be specified by including elements in the priors list. For both functions, μ has a normal prior, with mean muMean and standard deviation muSD. For Bnormal, a gamma prior is used for the precision, \( \tau = 1 \) \( \sigma^2 \), with parameters specified by tauShape and tauRate. For Bnormal2, a gamma prior is placed on \( \sigma \), with parameters specified by mode, sigmaMode, and SD, sigmaSD.

When priors = NULL (the default), Bnormal uses improper flat priors for both \( \mu \) and \( \tau \), while Bnormal2 uses a broad normal prior (muMean = mean(y), muSD = sd(y)*5) for \( \mu \) and a uniform prior on \( (sd(y) / 1000, sd(y) * 1000) \) for \( \sigma \).

Value

Returns an object of class mcmcOutput.

Author(s)

Mike Meredith, Bnormal based on code by Brian Neelon, Bnormal2 adapted from code by John Kruschke.
References


Examples

# Generate a sample from a normal distribution, maybe the head-body length of a # carnivore in mm:
HB <- rnorm(10, 900, 15)
Bnormal(HB) # with improper flat priors for mu and tau
Bnormal(HB, priors=list(muMean=1000, muSD=200))
Bnormal(HB, priors=list(muMean=1, muSD=0.2)) # a silly prior produces a warning.

if(requireNamespace("rjags")) {
  Bnormal2(HB, chains=2) # with broad normal prior for mu, uniform for sigma
  Bnormal2(HB, chains=2, priors=list(muMean=1000, muSD=200, sigmaMode=20, sigmaSD=10))
}

Bayesian Occupancy Single Season

Bayesian single-season occupancy modelling

Description

Functions to estimate occupancy from detection/non-detection data for a single season using a Gibbs sampler coded in R or JAGS.

BoccSS0 runs a model in R without covariates, and allows priors to be specified as beta distributions for probability of occupancy and probability of detection.

BoccSS runs a model in R allowing for covariates, using a probit link and conjugate normal priors, which can be specified as mean and covariance.

Usage

BoccSS0(y, n, psiPrior=c(1,1), pPrior=c(1,1),
  chains=3, draws=30000, burnin=100, ...)

BoccSS(DH, model=NULL, data=NULL, priors=list(),
  chains=3, draws=30000, burnin=1000, thin=1, parallel,
  seed=NULL, doWAIC=FALSE, ...)
Bayesian Occupancy Single Season

Arguments

- **y**
  a vector with the number of detections at each site; or a 1/0/NA matrix (or data frame) of detection histories, sites x occasions.

- **n**
  a scalar or vector with the number of visits (survey occasions) at each site; ignored if y is a matrix or data frame.

- **psiPrior, pPrior**
  parameters for beta distributions to be used as priors for psi and p.

- **DH**
  a 1/0/NA matrix (or data frame) of detection histories, sites x occasions.

- **model**
  a list of formulae symbolically defining a linear predictor for each parameter in terms of covariates. If NULL, an intercept-only model is used, ie, psi(.) p(.).

- **data**
  a data frame containing the variables in the model. For occSStime, a data frame with a row for each survey occasion; otherwise, a row for each site. Each site covariate has one column. Each survey covariate has one column for each occasion, and the column name must end with the occasion number (without leading zeros); eg, Cov1, Cov2, ..., Cov15. All covariates should be included in data, otherwise they will be sought in enclosing environments, which may not produce what you want – and they won’t be standardised.

- **priors**
  a list with elements for prior mean and variance for coefficients; see Details. If NULL, improper flat priors are used.

- **chains**
  number of MCMC chains to run.

- **draws**
  minimum number of values to return. The actual number will be a multiple of the number of chains.

- **burnin**
  number of iterations per chain to discard as burn-in.

- **thin**
  the thinning interval between consecutive values in the chain.

- **parallel**
  logical; if TRUE and n.chains > 1 and available cores (as returned by parallel::detectCores) > 2, chains will be run in parallel. If missing, chains will be run in parallel if n.chains < available cores.

- **doWAIC**
  logical; if TRUE, the Watanabe-Akaike Information Criterion is calculated. NOTE: THIS FEATURE IS STILL EXPERIMENTAL.

- **seed**
  for reproducible results; note that parallel and sequential methods use different random number generators, so will give different results with the same seed.

Details

BoccSS0 implements a simple model with one parameter for probability of occupancy and one for probability of detection, ie. a psi(.) p(.) model, using a Gibbs sampler implemented in R.

Independent beta distributions are used as priors for BoccSS0, as specified by psiPrior and pPrior. The defaults, c(1,1), correspond to uniform priors on the probabilities of occupancy and detection.

BoccSS uses a probit link to model occupancy and detection as a function of covariates (Dorazio and Rodriguez 2011); most software uses a logistic (logit) link. See Links. Coefficients on the probit scale are about half the size of the equivalent on the logit scale.

Priors for BoccSS are listed in the priors argument, which may contain elements:
muPsi and muP: the means for occupancy and detection coefficients respectively. This may be a vector with one value for each coefficient, including the intercept, or a scalar, which will be used for all. The default is 0.

sigmaPsi and sigmaP: the (co)variance for occupancy and detection coefficients respectively. This may be (1) a vector with one value for each coefficient, including the intercept, which represents the variance, assuming independence, or (2) a scalar, which will be used for all, or (3) a variance-covariance matrix. The default is 1, which for a probit link and standardized covariates is only mildly informative.

When specifying priors, note that numerical covariates are standardized internally before fitting the model. For an intercept-only model, a prior of Normal(0, 1) on the probit scale implies a Uniform(0, 1) or Beta(1, 1) prior on the probability scale.

If you are unsure of the order of predictors, do a short run and check the output, or pass unusable values (e.g., muPsi=numeric(100)) and check the error message.

Value

Returns an object of class mcmcOutput.

Author(s)

Mike Meredith. BoccSS uses the Gibbs sampler described by Dorazio and Rodriguez (2012).

References


See Also

See the examples for the weta data set.

Examples

# The blue ridge salamanders data from MacKenzie et al (2006) p99:
data(salamanders)
y <- rowSums(salamanders)
n <- rowSums(!is.na(salamanders))

tmp <- BoccSS0(y, n)
tmp
tmp
occSS0(y, n) # for comparison
plot(tmp)
**Bayesian Poisson analysis**

*Bayesian analysis of count data*

---

**Description**

Generates random draws from the posterior for a Poisson likelihood and gamma prior.

**Usage**

```r
Bpoisson(y, n, priors=NULL, draws=10000, ...)
```

**Arguments**

- `y` the count
- `n` the sample size
- `priors` an optional list with elements specifying the priors for the mode and SD of the gamma prior distribution; see `Details`.
- `draws` the number of MCMC draws to be returned.
- `...` additional arguments to pass to the function.

**Details**

The function generates a vector of random draws from the posterior distribution of the probability of the observed count. It uses conjugacy to determine the parameters of the posterior gamma distribution, and draws independent values from this.

A prior can be specified with the `priors` argument. A gamma prior is used, specified by mode, `mode`, and SD, `sd`.

When `priors = NULL` (the default), a uniform prior corresponding to `gamma(1, 0)` is used.

**Value**

Returns an object of class `mcmcOutput`.

**Author(s)**

Mike Meredith.

**Examples**

```r
# Generate a sample from a Poisson distribution, maybe the number of ticks observed on a sample of rodents:
# n <- 10 # number of trials (rodents examined)
( y <- rpois(n, 1.2) ) # number of ticks on each rodent
Bpoisson(sum(y), n) # with uniform prior
plot(Bpoisson(sum(y), n))
Bpoisson(sum(y), n, priors=list(mode=1, sd=3)) # with informative prior
plot(Bpoisson(sum(y), n, priors=list(mode=1, sd=3)))
```
Bayesian SCR

Spatially explicit capture-recapture (secr) density estimation using MCMC

Description

Functions to estimate density from mark-recapture data using MCMC methods and JAGS.

Usage

Bsecr0(capthist, buffer = 100, start = NULL, nAug = NA, maxSig = 2*buffer, 
chains=3, draws=1e4, burnin=0, thin=1, adapt=1000, 
priorOnly=FALSE, parallel=NULL, seed=NULL, ...)

Arguments

capthist   a capthist object as defined in package secr including capture data and detector (trap) layout
buffer     scalar mask buffer radius (default 100 m)
start      an optional object of class secr, ie, output from the secr.fit function in package secr; objects of other classes are silently ignored.
nAug       number of individuals in the augmented population; if NA, a suitable default is chosen based on the object passed to start or a preliminary run of secr.fit.
maxSig     maximum value for the scale parameter of the detection function: the prior is Uniform(0, maxSig).
chains     the number of Markov chains to run.
draws      the total number of values to return. The number of values calculated per chain is adapt + burnin + ceiling(draws / chains) * thin.
burnin     the number of values to discard at the beginning of each chain.
thin       the thinning rate. If set to n > 1, n values are calculated for each value returned.
adapt      the number of iterations to run in the JAGS adaptive phase.
priorOnly  if TRUE, the function produces random draws from the appropriate prior distributions, with a warning.
seed       set a seed for the random number generators.
parallel   if TRUE or NULL and sufficient cores are available, the MCMC chains are run in parallel; if TRUE and insufficient cores are available, a warning is given.

Details

Bsecr0 implements an intercept-only model (D ~ 1, g0 ~ 1, sigma ~ 1).
BetaDist

Value

Returns an object of class Bwiqid, data frame with one column for each parameter, ie. D, lam0 and sigma.

There are print, plot, and window methods for Bwiqid.

Author(s)

Mike Meredith

References


See Also

The function secr.fit in package secr.

Examples

```r
if(requireNamespace("secr") && requireNamespace("rjags")) {
  # The stoats data set in 'secr'
  data(stoatDNA, package="secr")
  # This takes ca 10 mins on a multicore machine:
  Bout <- Bsecr0(stoatCH, buffer=1000, chains=2) # 2 chains for testing
  summary(Bout)
  plot(Bout)
  # look at diagnostic plots to see if D is constrained by nAug:
  diagPlot(Bout) # Upper values of D doesn't look constrained.
  plotACs(Bout, 1:20) # Plot the ACs for captured animals
}
```

The Beta Distribution

Description

Density, distribution function, quantile function and random generation for the Beta distribution with parameters mean and sd OR mode and concentration. These are wrappers for stats::dbeta, etc. getBeta*Par returns the shape parameters.
Usage

dbeta2(x, mean, sd)
pbeta2(q, mean, sd, lower.tail=TRUE, log.p=FALSE)
qbeta2(p, mean, sd, lower.tail=TRUE, log.p=FALSE)
rbeta2(n, mean, sd)
getBeta2Par(mean, sd)

dbeta3(x, mode, concentration)
pbeta3(q, mode, concentration, lower.tail=TRUE, log.p=FALSE)
qbeta3(p, mode, concentration, lower.tail=TRUE, log.p=FALSE)
rbeta3(n, mode, concentration)
getBeta3Par(mode, concentration)

Arguments

x vector of parameter values
q vector of quantiles
p vector of probabilities
n number of random draws required.
mean mean of the beta distribution; cannot be 0 or 1.
sd standard deviation of the beta distribution; this must be less than \( \sqrt{mean \times (1-mean)} \), larger values will return NA, with a warning.
mode mode of the beta distribution; may be 0 or 1.
concentration concentration of the beta distribution; concentration = 2 is uniform, and the distribution becomes narrower as concentration increases. It is sometimes referred to as 'sample size', but best thought of as sample size + 2.
lower.tail logical; if TRUE (default), cumulative probabilities up to x, otherwise, above x.
log.p logical; if TRUE, probabilities p are given as log(p).

Value

dbeta2 gives the density, pbeta2 gives the distribution function, qbeta2 gives the quantile function, and rbeta2 generates random deviates.
getBeta2Par returns a 2-column matrix with the shape parameters corresponding to mean and sd OR mode and concentration.

Author(s)

Mike Meredith

See Also

See the stats functions dbeta, pbeta, qbeta, rbeta.
Examples

# Plot some curves with dbeta2
xx <- seq(0, 1, length.out=101)
plot(xx, dbeta2(xx, 0.4, 0.12), xlab="x", ylab="Probability density",
main="Beta curves with mean = 0.4", type='l', lwd=2)
lines(xx, dbeta2(xx, 0.4, 0.24), col='darkgreen', lwd=2)
lines(xx, dbeta2(xx, 0.4, 0.28), col='red', lwd=2)
lines(xx, dbeta2(xx, 0.4, 0.36), col='blue', lwd=2)
abline(v=0.4, lty=3, lwd=2)
legend('topright', paste("sd =", c(0.12,0.24, 0.28, 0.36)), lwd=2,
col=c('black', 'darkgreen', 'red', 'blue'), bty='n')

# Get shape and rate parameters for mean = 0.4 and sd = c(0.12,0.24, 0.28, 0.36, 0.49)
# The last value for sd is too big and will produce NAs and a warning
getBeta2Par(mean = 0.4, sd = c(0.12,0.24, 0.28, 0.36, 0.49))

# The parameterisation with mean and sd doesn't seem intuitive,
# let's try mode and concentration.
plot(xx, dbeta3(xx, 0.4, 16), xlab="x", ylab="Probability density",
main="Beta curves with mode = 0.4", type='l', lwd=2)
lines(xx, dbeta3(xx, 0.4, 8), col='darkgreen', lwd=2)
lines(xx, dbeta3(xx, 0.4, 4), col='red', lwd=2)
lines(xx, dbeta3(xx, 0.4, 2), col='blue', lwd=2)
abline(v=0.4, lty=3, lwd=2)
legend('topleft', paste("concentration =", c(16, 8, 4, 2)), lwd=2,
col=c('black', 'darkgreen', 'red', 'blue'), bty='n')

# The mode can be at 0 or 1:
plot(xx, dbeta3(xx, 1, 16), xlab="x", ylab="Probability density",
main="Beta curves with mode = 1", type='l', lwd=2)
lines(xx, dbeta3(xx, 1, 8), col='darkgreen', lwd=2)
lines(xx, dbeta3(xx, 1, 4), col='red', lwd=2)
lines(xx, dbeta3(xx, 1, 2), col='blue', lwd=2)
legend('topleft', paste("concentration =", c(16, 8, 4, 2)), lwd=2,
col=c('black', 'darkgreen', 'red', 'blue'), bty='n')

# Cumulative plots with pbeta3
plot(xx, pbeta3(xx, 0.4, 16), xlab="x", ylab="Cumulative probability",
main="Beta curves with mode = 0.4", type='l', lwd=2)
lines(xx, pbeta3(xx, 0.4, 8), col='darkgreen', lwd=2)
lines(xx, pbeta3(xx, 0.4, 4), col='red', lwd=2)
lines(xx, pbeta3(xx, 0.4, 2), col='blue', lwd=2)
abline(v=0.4, lty=3, lwd=2)
legend('topleft', paste("concentration =", c(16, 8, 4, 2)), lwd=2,
col=c('black', 'darkgreen', 'red', 'blue'), bty='n')

# Generate random draws and plot a histogram
rnd <- rbeta3(1e5, 0.4, 8)
hist(rnd, freq=FALSE)
# Add the curve:
lines(xx, dbeta3(xx, 0.4, 8), col='darkgreen', lwd=2)
# Get shape and rate parameters for mode = 0.4 and concentration = c(2, 4, 8, 16)
getBeta3Par(mode = 0.4, concentration = c(2, 4, 8, 16))

Bwiqid-class                  Conversion to class Bwiqid (deprecated)

Description
Convert output containing MCMC chains to the class mcmcOutput, with a warning. The class Bwiqid is deprecated, use mcmcOutput instead.

Usage
as.Bwiqid(object, ...)

## Default S3 method:
as.Bwiqid(object, ...)

Arguments

object an object containing the MCMC chains; see Details.

... named parameters to be passed to other methods.

Value
An object of class mcmcOutput.

Author(s)
Mike Meredith.

Closed Captures                  Analysis of mark-recapture data for closed populations

Description
Functions to analyse the classical models for closed populations without individual covariates, ie. full likelihood models.
Closed Captures

Usage

closedCapM0(CH, ci = 0.95, ciType=c("normal", "MARK"), ...)
closedCapMb(CH, ci = 0.95, ciType=c("normal", "MARK"), ...)
closedCapMt(CH, ci = 0.95, ciType=c("normal", "MARK"), ...)
closedCapMtcov(CH, model=list(p~1), data=NULL, ci = 0.95, ciType=c("normal", "MARK"), ...)
closedCapMh2(CH, ci = 0.95, ciType=c("normal", "MARK"), ...)
closedCapMhJK(CH, ci = 0.95)

Arguments

CH a 0/1 capture history matrix, animals x occasions; NAs not permitted. For functions closedCapM0, closedCapMh2 and closedCapMhJK, CH can be a vector of capture frequencies of length equal to the number of occasions - trailing zeros are required.

model a list of formulae symbolically defining a linear predictor for p in terms of covariates.

data a data frame containing the variables in the model, with one row for each occasion.

ci the required confidence interval.

ciType the method used to calculate the confidence interval for population size (N); see Details.

... other arguments passed to nlm.

Details

Model M0 assumes all animals have the same capture probability.

Model Mb allows for a permanent behavioural response: capture probability is different for animals that have already been captured at least once.

Model Mh2 and the jackknife estimator allow for heterogeneity in capture probability.

The likelihood maximization routine produces an estimate and standard error for beta = log(f0), where f0 is the number of animals never captured. If ciType == "normal", a confidence interval for beta is calculated as beta +/- crit * SE.beta, where crit is the appropriate multiplier for the confidence interval required, 1.96 for a 95% CI. This confidence interval is then back-transformed to the real scale and added to the number of animals captured (M[t+1]) to give estimates of N.

If ciType == "MARK", the method used by MARK to calculate confidence intervals is used (see MARK help page for Closed Capture Models and Burnham et al (1987, p212)). The beta values are back-transformed with f0.hat = exp(beta) and SE.f0 = SE.beta * f0.hat, and hence CV = SE.f0 / f0.hat. The confidence limits are then
Lower = \( f_0 \cdot \text{hat} / C + M[t+1] \)
Upper = \( f_0 \cdot \text{hat} \times C + M[t+1] \)

where \( C = \exp(\text{crit} \times \sqrt{\log(1 + CV^2)}) \).

Confidence intervals for capture probabilities are always calculated on the logit scale and back-transformed to real values.

**Value**

Returns an object of class `wiqid`, which is a list with the following elements:

- `call`: The call used to produce the results
- `beta`: Values of the coefficients of the terms in the linear predictors, with standard errors and confidence intervals.
- `beta.vcv`: The variance-covariance matrix for the beta estimates.
- `real`: Estimates of population size (N) and probability of detection on the real scale, with confidence intervals.
- `logLik`: A vector with elements for log(likelihood), number of parameters, and effective sample size. If the variance-covariance matrix cannot be calculated, the second element should be `NA`.

The jackknife estimator does not use likelihood maximisation, so elements `beta` and `beta.vcv` are `NULL` and `logLik = NA`.

There are `print`, `logLik`, and `nobs` methods for class `wiqid`.

**Author(s)**

Mike Meredith

**References**

Basic work on mark-recapture for closed populations is in:


White, G C; D R Anderson; K P Burnham; D L Otis. 1982. *Capture-recapture and removal methods for sampling closed populations*. Los Alamos National Laboratory, Los Alamos NM.

Calculation of the confidence interval for N is in:


The jackknife estimator is described in:


Data sets in the examples are from:
White et al, op. cit.

Examples

# Data from White et al (1982):
freq1 <- c(50, 46, 35, 24, 14, 5, 0) # there were 7 capture occasions
closedCapM0(freq1)
closedCapM0(freq1, ci=0.8)
closedCapMh2(freq1)
closedCapMhJK(freq1)

data(KanhaTigers)
closedCapM0(KanhaTigers)
closedCapMb(KanhaTigers)
closedCapMh2(KanhaTigers)
closedCapMhJK(KanhaTigers)
closedCapMt(KanhaTigers)
closedCapMtcov(KanhaTigers, p~.Time)
# Generate some mythical covariates:
covars <- data.frame(Temp = runif(ncol(KanhaTigers), 15, 25),
                      Cloud = sample(0:8, ncol(KanhaTigers), replace=TRUE))
closedCapMtcov(KanhaTigers, p~Cloud, data=covars)

# Compare the normal (default) and MARK confidence intervals for N:
rbind(closedCapMt(KanhaTigers)$real[1, ],
closedCapMt(KanhaTigers, ciType="MARK")$real[1, ])

---

**deprecated**

*Deprecated functions in package wiqid*

**Description**

These functions have been replaced by functions in `mcmcOutput`

**Usage**

`plotPost(...)`

**Arguments**

... arguments passed to the new function.

**Details**

`plotPost` has been replaced by `postPlot`. 
Value
See help for the new function.

Author(s)
Mike Meredith

---

**dippers**

*Capture-recapture data for European dippers*

Description
A data set that accompanies Program MARK and is included in the RMark package in a different format under the name dipper.

Usage
data(dippers)

Format
A data frame with 294 observations on the following 8 variables.

Y1, Y2, Y3, Y4, Y5, Y6, Y7  detection histories for 294 dippers over 7 years: '1' if captured, '0' if not captured.

sex  sex of each bird captured.

Source

References
Analysis given in many books and papers, notably:

Examples
data(dippers)

DH <- dippers[1:7]  # Extract the detection histories
survCJS(DH)  # the phi(.) p(.) model
survCJS(DH, phi ~ .time)  # the phi(t) p(.) model

# Floods affected the 2nd and 3rd intervals
Distance Measures

df <- data.frame(flood = c(FALSE, TRUE, TRUE, FALSE, FALSE, FALSE))
survCJS(DH, phi ~ flood, data=df)

# Including a grouping factor:
survCJS(DH, phi ~ flood * group, data=df, group=dippers$sex)

# Bayesian estimation:

if(requireNamespace("rjags")) {
  Bdip <- BsurvCJS(DH, parallel=FALSE)
  plot(Bdip)
  BdipFlood <- BsurvCJS(DH, list(phi ~ flood, p ~ .time), data=df, parallel=FALSE)
  BdipFlood
  op <- par(mfrow=2:1)
  plot(BdipFlood, "phi[1]", xlim=c(0.3, 0.75), main="No flood")
  plot(BdipFlood, "phi[2]", xlim=c(0.3, 0.75), main="Flood")
  par(op)
  ratio <- BdipFlood['phi[2]'] / BdipFlood['phi[1]']
  postPlot(ratio, compVal=1)
}

Distance Measures

Plug-in distance-measure functions for distShell.

Description

Each function takes two (interchangeable) vectors of data and returns a measure of distance between them. Vectors may be just 1/0 values (presence-absence data) or non-negative integers (count data).

Usage

distBrayCurtis(d1, d2)
distChaoJaccCorr(d1, d2)
distChaoJaccNaive(d1, d2)
distChaoSorCorr(d1, d2)
distChaoSorNaive(d1, d2)
distChord(d1, d2)
distJaccard(d1, d2)
distMatching(d1, d2)
distMorisitaHorn(d1, d2)
distOchiai(d1, d2)
distPreston(d1, d2)
distRogersTanimoto(d1, d2)
distSimRatio(d1, d2)
distSorensen(d1, d2)
distWhittaker(d1, d2)
Distance Measures

Arguments

\[ d_1, d_2 \]

vectors of equal length, specifying the two cases to be compared.

Details

**distBrayCurtis** Complement of the Bray-Curtis index, see Magurran p.246, where it is referred to as the 'quantitative Sorensen' index. Based on count data.

**distChaoJaccCorr, distChaoJaccNaive, distChaoSorCorr, distChaoSorNaive** Each is the complement of one of a series of similarity indices which allow for (1) relative abundance of shared species and (2) estimation of number of shared species not detected. Based on count data. See Chao et al. 2005.

**distChord** Both vectors are normalized so that the sum of squares = 1, i.e. they lie on the surface of a sphere of unit radius. The distance measure is the length of the cord joining the two points through the sphere, which is in \([0, \sqrt{2}]\), i.e. it is \(\sqrt{2}\) for sites with no species in common. Based on count data. See Zuur et al 2007:166, Legendre & Legendre 1998:279.

**distJaccard** Complement of the Jaccard index of similarity; also known as "Marczewski-Steinhaus distance". Based on presence-absence data. No. of shared species / Overall number of species.

**distMatching** A simple matching index: the proportion of elements which match in two presence-absence vectors (i.e. present in both or absent in both). Zuur et al 2007:165.

**distMorisitaHorn** The complement of the Morisita-Horn index of similarity. Based on count data. See Magurran 2004:246 The Morisita-Horn index is also known as "simplified Morisita". The "Morisita" and "Horn" indices are different again! See Krebs 1999:470-471.


**distPreston** Preston’s coefficient of faunal dissimilarity (z). Based on presence-absence data. See Preston 1962:418.

**distRogersTanimoto** Complement of Rogers and Tanimoto’s coefficient of similarity. Based on presence-absence data. See Zuur et al 2007:165.


**distSorensen** Complement of Sorensen (or Dice) index of similarity. Based on presence-absence data. No. of shared species / Average number of species. Same as Whittaker’s distance measure for Incidence (presence-absence) data minus one (Magurran 2004:244).


Value

a scalar, the distance between the two vectors.

Author(s)

Mike Meredith
Distance Measures

References


See Also

The basic distance computation function is `dist` in package stats. Other functions are `vegan::vegdist` and `labdsv::dsvdis`.

These functions provide the following distance measures:

• binary (in dist) = asymmetric binary = Steinhaus

• binomial (in vegdist)

• bray/curtis (in dsvdis) = bray (in vegdist)

• canberra (in dist and vegdist)

• chao (in vegdist)

• chisq (in dsvdis)

• euclidean (in dist and vegdist)

• gower (in vegdist)

• horn (in vegdist) = Morisita-Horn or simplified Morisita

• jaccard (in vegdist)

• kulczynski (in vegdist)

• manhattan (in dist and vegdist)

• maximum (in dist)

• minkowski (in dist)

• morisita (not simplified!) (in vegdist)

• mountford (in vegdist)

• ochiai (in dsvdis)

• raup (in vegdist) = Raup-Crick

• roberts (in dsvdis)

• ruzicka (in dsvdis)

• sorenson (in dsvdis)

• steinhaus (in dsvdis)= binary
**Examples**

```r
data(distTestData)
distShell(distTestData, distJaccard)
distShell(distTestData, distMorisitaHorn)
```

**distShell**  
*Distance Matrix Computation*

**Description**

 Produces a 'dist' object using a user-defined distance measure.

**Usage**

```r
distShell(DATA, FUNC, diag = FALSE, upper = FALSE, ...)
```

**Arguments**

- **DATA**  
  a matrix-like object with variables in COLUMNS, cases in ROWS.
- **FUNC**  
  the distance function; takes two vector arguments and returns a single scalar distance measure. See Details.
- **diag**  
  logical value indicating whether the diagonal of the distance matrix should be printed by print.dist.
- **upper**  
  logical value indicating whether the upper triangle of the distance matrix should be printed by print.dist.
- **...**  
  further arguments, passed to FUNC.

**Details**

FUNC must be a function of the form foo(x1, x2, ...). The first two arguments must be vectors of equal length, specifying the two cases to be compared. It must return a single scalar distance measure. Similarity measures will work, but for consistency stick to distance measures.

A number of example functions are provided in the package; see Distance Measures.

**Value**

distShell returns an object of class "dist", including the attribute call.

See dist for details of this class.

**Author(s)**

Mike Meredith, 10 Dec 2006, updated 1 Sept 2012.
distTestData

See Also

dist in package stats. Also vegan::vegdist and labdsv::dsvdis. See Distance Measures for details of plug-in functions.

Examples

# Use the artificial data set, see ?distTestData
data(distTestData)

# Using distance measure functions in this package:
distShell(distTestData, distSorensen)
distShell(distTestData, distMorisitaHorn)

# Write a customised distance function:
K <- function(a1, a2) {
  shared <- sum(a1 > 0 & a2 > 0)
  notshared <- sum(xor(a1 > 0, a2 > 0))
  shared / notshared
}
distShell(distTestData, K)
# This returns Inf if the number of species not shared is zero. May not be a good measure!

---

distTestData  An artificial data set to test distance/dissimilarity measures

Description

Artificial data for counts of 32 species at 5 sites.

Usage

data(distTestData)

Format

A matrix with 5 rows (sites), labelled A-B, and 32 columns (species).

Details

Sites A, B and C each have 16 species and 158 individuals.
Sites A and B have the same species, but the numbers of each are different.
Site C has a completely different set of 16 species, but the same number of individuals.
Site D has the same species in the same proportions as A, but twice the number of individuals.
Site E has 32 species and 316 individuals.

Source

Artificial data.
Examples

data(distTestData)
# Display the data:
print(t(distTestData))

distShell(distTestData, distJaccard)
# A B C D
# B 0.0
# C 1.0 1.0
# D 0.0 0.0 1.0
# E 0.5 0.5 0.5 0.5
# Jaccard index ignores counts, so sees AB, AD and BD as identical (zero distance).

round(distShell(distTestData, distMorisitaHorn), 2)
# A B C D
# B 0.89
# C 1.00 1.00
# D 0.00 0.89 1.00
# E 0.33 0.93 0.33 0.33
# Morisita-Horn index considers proportions, so AD are identical but not AB or BD.

round(distShell(distTestData, distBrayCurtis), 2)
# A B C D
# B 0.84
# C 1.00 1.00
# D 0.33 0.84 1.00
# E 0.33 0.89 0.33 0.50
# Bray-Curtis index is affected by abundance as well as proportions, so AD are no longer identical.
# Site C only overlaps with D, so AC, BC and CD are 1.00 for all indices.
# Site E overlaps with all the others, so AE, BE, CE and DE all lie between 0 and 1 for all indices.

---

**Diversity indices**  **Biodiversity indices**

Description

Common indices of biodiversity, expressed as the number of common species.

Usage

biodSimpson(abVec, correct = TRUE)
biodShannon(abVec)
biodBerger(abVec)
biodBrillouin(cntVec)

Arguments

abVec  a vector of measures of abundance, eg. counts of individuals or biomass, one element per species; or a corresponding matrix or data frame, which will be converted to a vector with rowSums.
Diversity indices

- **cntVec**: a vector (or matrix or data frame) of counts of individuals, one element per species. Non-integers will be rounded without warning.
- **correct**: if TRUE, a small sample correction is applied, and in that case abVec should have count data (non-integers will be silently rounded).

**Details**

- **biodSimpson**: Inverse of Simpson’s (1949) index of dominance. If `correct` = TRUE, a small-sample correction is applied, giving Hurlbert’s (1971) diversity index. Otherwise, the result is equivalent to Hill’s (1973) \( N_2 \).
- **biodShannon**: Exponential form of Shannon’s (1948) entropy measure, equivalent to Hill’s (1973) \( N_1 \).
- **biodBerger**: Inverse of Berger & Parker’s (1970) index of dominance, equivalent to Hill’s (1973) \( N_{inf} \).
- **biodBrillouin**: Exponential form of Brillouin’s index: for small, completely censused populations, Brillouin’s index is a more appropriate measure of entropy than Shannon’s measure (Maurer & McGill 2011:61).

**Value**

The relevant index.

**Warning**

It is important that the proportions of each species in the sample represent those in the population from which it is drawn. This will not be the case if probability of inclusion varies among species, as often occurs when samples are collected in the field.

**Author(s)**

Mike Meredith

**References**

See Also

richSobs and Species richness estimators for alternatives to indices.

Examples

data(KillarneyBirds)
apply(KillarneyBirds, 2, biodSimpson)

Description

Density, distribution function, quantile function and random generation for the Gamma distribution with parameters mean and sd. These are wrappers for stats::dgamma, etc. getGammaPar returns the shape and rate parameters.

Usage

dgamma2(x, mean, sd)
pgamma2(q, mean, sd, lower.tail=TRUE, log.p=FALSE)
qgamma2(p, mean, sd, lower.tail=TRUE, log.p=FALSE)
rgamma2(n, mean, sd)
getGammaPar(mean, sd)

Arguments

x vector of parameter values
q vector of quantiles
p vector of probabilities
n number of random draws required.
mean mean of the gamma distribution
sd standard deviation of the gamma distribution
lower.tail logical; if TRUE (default), cumulative probabilities up to x, otherwise, above x.
log.p logical; if TRUE, probabilities p are given as log(p).

Value

dgamma2 gives the density, pgamma2 gives the distribution function, qgamma2 gives the quantile function, and rgamma2 generates random deviates.

getGammaPar returns a 2-column matrix with the shape and rate parameters corresponding to mean and sd.
getMCerror

Author(s)

Mike Meredith

See Also

See the stats functions dgamma, pgamma, qgamma, rgamma.

Examples

# Plot some curves with dgamma2
xx <- seq(0, 20, length.out=101)
plot(xx, dgamma2(xx, 5, 1), xlab="x", ylab="Probability density",
     main="Gamma curves with mean = 5", type=’l’, lwd=2)
lines(xx, dgamma2(xx, 5, 2), col=’darkgreen’, lwd=2)
lines(xx, dgamma2(xx, 5, 4), col=’red’, lwd=2)
lines(xx, dgamma2(xx, 5, 8), col=’blue’, lwd=2)
abline(v=5, lty=3, lwd=2)
legend(’topright’, paste("sd =", c(1,2,4,8)), lwd=2,
       col=c(’black’, ‘darkgreen’, ‘red’, ‘blue’), byt=’n’)

# Cumulative plots with pgamma2
plot(xx, pgamma2(xx, 5, 1), xlab="x", ylab="Cumulative probability",
     main="Gamma curves with mean = 5", type=’l’, lwd=2)
lines(xx, pgamma2(xx, 5, 2), col=’darkgreen’, lwd=2)
lines(xx, pgamma2(xx, 5, 4), col=’red’, lwd=2)
lines(xx, pgamma2(xx, 5, 8), col=’blue’, lwd=2)
abline(v=5, lty=3, lwd=2)
legend(’bottomright’, paste("sd =", c(1,2,4,8)), lwd=2,
       col=c(’black’, ‘darkgreen’, ‘red’, ‘blue’), byt=’n’)

# Generate random draws and plot a histogram
rnd <- rgamma2(1e5, 5, 2)
hist(rnd, freq=FALSE)
# Add the curve:
lines(xx, dgamma2(xx, 5, 2), col=’darkgreen’, lwd=2)

# Get shape and rate parameters for mean = 5 and sd = c(1,2,4,8)
getGammaPar(mean = 5, sd = c(1,2,4,8))

getMCerror

MCMC error using the batch method (deprecated)

Description

This is now a wrapper for getMCE

Usage

getMCerror(object, n.chains, SDpc=FALSE)
Arguments

object  an object of any class with MCMC output that can be coerced to class mcmcOutput.
n.chains  ignored
SDpc  if TRUE, the value of the MC error as a percentage of the posterior SD will be returned.

Value

If SDpc is FALSE (the default), a named vector with the estimates of MC error. If TRUE, the MC error as a percentage of the standard deviation of the posterior chain. A value <5% of SD is adequate for most purposes, but <1.5% is needed to properly estimate tail probabilities (Lunn et al 2013, p78-79).

Author(s)

Mike Meredith

References


Examples

# Get some output to use
data(salamanders)
y <- rowSums(salamanders)
(out <- BoccSS0(y, 5))
getMCerror(out)
getMCerror(out, SDpc=TRUE)

GrandSkinks  Multi-season detection data for grand skinks

Description

Results of an occupancy survey of 352 rocky outcrops ("tors") looking for grand skinks. Tors were surveyed up to 3 times per year for 5 years. The surrounding terrain was characterised as natural grassland or pasture.

Usage

data(GrandSkinks)
KanhaTigers

Format

A data frame with 352 observations on the following 16 variables.

A1, A2, A3, B1, B2, B3, C1, C2, C3, D1, D2, D3, E1, E2, E3 an array of observations of detection (1) or nondetection (0) of skinks for each of 3 occasions in 5 years. NA indicates occasions when a tor was not visited.

habitat a factor indicating the type of grassland surrounding the tor.

Details

The data are provided as a data frame, such as would result from reading in data from a text file. Further formatting is needed before using these for analysis: see the examples.

Source

Data distributed with PRESENCE.

References


Examples

data(GrandSkinks)

# Extract detection histories:
DH <- GrandSkinks[, 1:15]
occMS0(DH, 3)

KanhaTigers

Capture history matrix for camera-trapped tigers

Description

Capture history matrix for camera-trapped tigers

Usage

data(KanhaTigers)

Format

A matrix with 26 rows for animals trapped and 10 columns for the trapping occasions. KanhaTigers[i, j] = 1 if animal i was trapped on occasion j, zero otherwise.
Source


Examples

data(KanhaTigers)
dim(KanhaTigers)
closedCapMt(KanhaTigers)

Description

The number of territories held by breeding males in 9 blocks of woodland habitat in County Killarney, Ireland.

Usage

data(KillarneyBirds)

Format

A data frame with counts for 31 species in 9 blocks of habitat. Row names are the English species names.

- Oak1  first of 3 oak wood sites
- Oak2  second of 3 oak wood sites
- Oak3  third of 3 oak wood sites
- Yew   a mature yew wood
- Sitka  a Sitka spruce plantation
- Norway a Norway spruce plantation
- Mixed  a mixed broadleaf wood
- Patchy a wood with patches of broadleaf and conifer trees
- Swampy a swampy seasonally-flooded woodland

Source


References

Examples

```r
data(KillarneyBirds)
## number of species in each block of habitat:
colSums(KillarneyBirds > 0)
```

Description

Generalised linear models with binomial response variables ("logistic regression") use a link function to link the response on a (0,1) scale to a linear predictor on (-Inf, Inf). The canonical link is the logistic ("logit") function, which has some nice theoretical properties and can be interpreted as the log of the odds of success. Other links are available, notably the cumulative standard normal ("probit") link, which allows for Gibbs sampling with truncated normal distributions. For that reason, several of the Bayesian estimation functions in `wiqid` use the probit link.

The form of the logit and probit links are shown in the figure below.

![logit and probit links](image)

Both curves are symmetric, with probability = 0.5 when the linear predictor = 0. The probit curve is steeper, so coefficients in a probit regression will be smaller than those in a logit regression (by a factor of about 1.7).
MeadowVoles

*Robust design mark-recapture data for meadow voles*

**Description**

Data for adult male meadow voles *Microtus pennsylvanicus* from a live-trapping study at Patuxent Wildlife Research Center (Nichols et al 1984). Trapping was carried out for 5 consecutive nights each month for 6 months (June to December 1981).

**Usage**

data(MeadowVoles)

**Format**

A data frame with 171 observations on the following 31 variables.

- **A1, A2, A3, A4, A5, B1, B2, B3, B4, B5, C1, C2, C3, C4, C5, D1, D2, D3, D4, D5, E1, E2, E3, E4, E5, F1, F2, F3, F4, F5**

  A 1/0 array of capture data for voles for each of 5 occasions per month for 6 months.

- **freq**

  A column with -1/1, where -1 indicates that the animal was not released after the last recorded capture.

**Details**

The data are provided as a data frame, such as would result from reading in data from a text file. Further formatting is needed before using these for analysis: see the examples.

**Source**


**References**


**Examples**

```r
data(MeadowVoles)

# Extract detection histories:
DH <- MeadowVoles[, 1:30]
freq <- MeadowVoles$freq

survRD(DH, freq=freq, occsPerSeason=5)
```
**occ2sps**  
*Single-season two-species occupancy estimation*

**Description**

Estimates occupancy and probability of detection for two species, where one (dominant) species affects the occupancy or detection of the other (subordinate) species (see Richmond et al, 2010). The model has the following parameters:

- \( \psi_A \) probability of occupancy of species A
- \( \psi_{Ba} \) probability of occupancy of B if A is absent
- \( \psi_{BA} \) probability of occupancy of B if A is present
- \( p_A \) probability of detection of species A if B is absent
- \( r_A \) probability of detection of species A if both are present
- \( p_B \) probability of detection of species B if A is absent
- \( r_{Ba} \) probability of detection of species B if both are present but A was not detected
- \( r_{BA} \) probability of detection of species B if both are present and A was detected

**Usage**

```r
occ2sps(DHA, DHB, model=NULL, data=NULL, ci = 0.95, verify=TRUE)
```

**Arguments**

- **DHA**: a 1/0/NA matrix (or data frame) of detection histories, sites x occasions, for the dominant species.
- **DHB**: detection histories for the subordinate species in the same format as DHA.
- **model**: a list of formulae symbolically defining a linear predictor for any of the parameters in the model. The default, NULL, is equivalent to `list(\( \psi_A \)~1,\( \psi_{Ba} \)~1,\( p_A \)~1,\( p_B \)~1);` parameters not included in the list are given the following values: \( \psi_{BA} \leftarrow \psi_{Ba} \), \( r_A \leftarrow p_A \), \( r_{Ba} \leftarrow p_B \), \( r_{BA} \leftarrow r_{Ba} \).
- **data**: a data frame containing the variables in the model. If `data = NULL`, a faster algorithm is used, and any covariates in the model will be ignored.
- **ci**: the confidence interval to use.
- **verify**: if TRUE, the data provided will be checked.

**Value**

Returns an object of class `wiqid`, see `wiqid-class` for details.

**Benchmarks**

Output has been checked against output from PRESENCE (Hines 2006) v.5.5 for the `railSims` data set. Real values are the same to 4 decimal places, and AICs are the same.
Author(s)

Mike Meredith

References


See Also

See the example data set *railSims*. See *occSS* for single-season single-species occupancy estimation.

Examples

data(railSims)

# Extract the two detection histories
DHA <- railSims[, 1:3]
DHB <- railSims[, 4:6]

# Default model (no interaction)
occ2sps(DHA, DHB)

# Add a submodel for psiBA, so that psiBA and psiBa are separated:
occ2sps(DHA, DHB, model = psiBA ~ 1)

# Add covariates for psiA and psiBA; only display beta coefficients:
occ2sps(DHA, DHB, model = list(psiA ~ logArea, psiBA ~ reeds), data=railSims)$beta

# Model corresponding to the data generation model
occ2sps(DHA, DHB, list(psiA ~ logArea, psiBA ~ reeds, rBA ~ 1), data=railSims)$beta

Occupancy Multi-Season

*Multi-season occupancy estimation*

Description

Functions to estimate occupancy from detection/non-detection data for multiple seasons. *occMS* is the general purpose function; it allows for site-, season- and survey-level covariates, but it is slow. *occMScovSite* excludes survey-level covariates, but is fast. *occMStime* and *occMS0* are simpler and faster.
Usage

occMS0(DH, occsPerSeason, ci=0.95, verify=TRUE, ...)

occMStime(DH, occsPerSeason, model=NULL, data=NULL, ci=0.95, verify=TRUE, ...)

occMS(DH, occsPerSeason, model=NULL, data=NULL, ci=0.95, verify=TRUE, ...)

occMScovSite(DH, occsPerSeason, model=NULL, data=NULL, ci=0.95, verify=TRUE, ...)

Arguments

DH
a 1/0/NA matrix (or data frame) of detection histories, sites x occasions. Rows with all NAs are silently removed.

occsPerSeason
the number of survey occasions per season; either a scalar if the number of surveys is constant, or a vector with one element for each season.

model
a list of formulae symbolically defining a linear predictor for each parameter in terms of covariates. The default corresponds to an intercept-only model.

data
a data frame containing the variables in the model: one row for each season or between-season period for occMStime and one for each site for occMScovSite. Each survey covariate has one column for each occasion, and the column name must end with the occasion number (without leading zeros); eg, Cov1, Cov2, ..., Cov15.

ci
the confidence interval to use.

verify
if TRUE, the data provided will be checked.

...
other arguments passed to nlm.

Details

occMS0 implements a simple multi-season model with one parameter each for initial occupancy, colonisation, local extinction, and probability of detection, ie. a psi1(.) gamma(.) epsilon(.) p(.) model.

occMStime allows for between-season differences in colonisation, local extinction, and probability of detection, either with covariates given in data or the in-built covariates .interval (for colonisation or extinction, or .season (for detection).

occMScovSite allows for between-season differences in colonisation, local extinction, and probability of detection with the in-built covariate .season and for between-site differences with covariates defined in data.

occMS allows for survey-level covariates in addition to the above, and separate covariates for between-season colonisation and local extinction.

Value

Returns an object of class wiqid, see wiqid-class for details.
Benchmarks

Output has been checked against output from PRESENCE (Hines 2006) v.5.5 for the GrandSkinks data set. Real values are mostly the same to 4 decimal places, though there is occasionally a discrepancy of 0.0001. AICs are the same.

Author(s)

Mike Meredith

References


Examples

data(GrandSkinks)
DH <- GrandSkinks[, 1:15]
occSS0(DH, 3)
occMStime(DH, 3, model=list(gamma ~ .interval, epsilon~1, p~.season))
occMScovSite(DH, 3,
    model=list(psi1~habitat, gamma ~ .interval, epsilon=habitat, p~.season),
    data=GrandSkinks)

Description

Functions to estimate occupancy from detection/non-detection data for a single season. occSS is the general-purpose function, and occSStime provides plots of detection probability against time. occSS0 and occSScovSite are faster functions for simpler models with summarized data. See occSSrn for the Royle-Nichols model for abundance-induced heterogeneity in detection probability.
Occupancy Single Season

Usage

```r
occSS(DH, model=NULL, data = NULL, ci=0.95, link=c("logit", "probit"), verify=TRUE, ...)

occSSstime(DH, model=p~1, data=NULL, ci=0.95, plot=TRUE, link=c("logit", "probit"), verify=TRUE, ...)

occSS0(y, n, ci=0.95, link=c("logit", "probit"), ...)

occSScovSite(y, n, model=NULL, data = NULL, ci=0.95, link=c("logit", "probit"), ...)
```

Arguments

- **DH**: a 1/0/NA matrix (or data frame) of detection histories, sites x occasions.
- **model**: a list of formulae symbolically defining a linear predictor for each parameter in terms of covariates. If NULL, an intercept-only model is used, i.e., psi(.) p(.).
- **ci**: the confidence interval to use.
- **data**: a data frame containing the variables in the model. For `occSSstime`, a data frame with a row for each survey occasion; otherwise, a row for each site. Each site covariate has one column. Each survey covariate has one column for each occasion, and the column name must end with the occasion number (without leading zeros); e.g., `Cov1, Cov2, ..., Cov15`. All covariates should be included in `data`, otherwise they will be sought in enclosing environments, which may not produce what you want – and they won’t be standardised.
- **link**: the link function to use, either logit or probit; see Links.
- **verify**: if TRUE, the data provided will be checked.
- **plot**: if TRUE (default), draws a plot of probability of detection vs time.
- **y**: a vector with the number of detections at each site.
- **n**: a scalar or vector with the number of visits (survey occasions) at each site.
- **...**: other arguments passed to `nlm`.

Details

`occSS` allows for psi or p to be modelled as a logistic or probit function of site covariates or survey covariates, as specified by `model`. It includes a built-in .time covariate which can be used for modelling p with time as a fixed effect, and .Time, .Time2, .Time3 for a linear, quadratic or cubic trend. A built-in .b covariate corresponds to a behavioural effect, where detection depends on whether the species was detected on the previous occasion or not.

`occSSstime` allows for time-varying covariates that are the same across all sites, e.g., moon-phase. Time variables are built in, as for `occSS`. A plot of detection probability vs time is produced if `plot=TRUE`.

`occSS0` implements a simple model with one parameter for probability of occupancy and one for probability of detection, i.e., a `psi(.) p(.)` model.

`occSScovSite` allows for site covariates but not for occasion or survey covariates.
Numeric covariates in data are standardised to facilitate convergence. This applies to binary covariates coded as 1/0; if this is not what you want, code these as TRUE/FALSE or as factors.

For speed, use the simplest function which will cope with your model. For example, you can run psi(.) p(.) models in occSScovSite or occSS, but occSS0 is much faster.

Value

Returns an object of class wiqid, see wiqid-class for details.

Benchmarks

Output has been checked against output from PRESENCE (Hines 2006) v.5.5 for the salamanders and weta data sets. Real values are mostly the same to 4 decimal places, though there is occasionally a discrepancy of 0.0001. AICs are the same.

Author(s)

Mike Meredith

References


See Also

See the examples for the weta data set. See occ2sps for single-season two-species models and occMS for multi-season models.

Examples

# The blue ridge salamanders data from MacKenzie et al (2006) p99:
data(salamanders)
occSS(salamanders)
occSStime(salamanders, p ~ .time) # time as a fixed effect
occSStime(salamanders, p ~ .Time + .Time2) # a quadratic time effect
occSS(salamanders, p ~ .b)

# or use the fast functions with y, n format:
y <- rowSums(salamanders)
n <- rowSums(!is.na(salamanders))
occSS0(y, n)
occSScovSite(y, n)
plot.Bwiqid

Plot method for objects of class 'Bwiqid'

Description

Method to display a plot showing the posterior probability distribution of one of the parameters of interest.

Usage

## S3 method for class 'Bwiqid'
plot(x, which=NULL, credMass=0.95, 
     ROPE=NULL, compVal=NULL, showCurve=FALSE, 
     showMode=FALSE, shadeHDI=NULL, ...)

Arguments

- **x**: an object of class Bwiqid.
- **which**: character; indicates which parameter to plot. If NULL and x has a defaultPlot attribute, that parameter is plotted; otherwise the parameter in column 1 is plotted.
- **credMass**: the probability mass to include in credible intervals; NULL suppresses plotting.
- **ROPE**: a two element vector, such as c(-1,1), specifying the limit of the ROPE on the estimate; see Details.
- **compVal**: a value for comparison with the parameter.
- **showCurve**: logical: if TRUE, the posterior density will be represented by a kernel density function instead of a histogram.
- **showMode**: logical: if TRUE, the mode of the posterior density will be shown instead of the mean.
- **shadeHDI**: specifies a colour to shade the area under the curve corresponding to the HDI; NULL for no shading. Ignored if showCurve = FALSE. Use colours() to see a list of possible colours.
- **...**: other graphical parameters.

Details

The posterior distribution is shown as a histogram or density curve (if showCurve = TRUE), together with the Highest Density Interval. A ROPE and comparison value are also shown if appropriate.

The probability that a parameter precisely zero (or has any other point value) is zero. More interesting is the probability that the difference from zero may be too small to matter. We can define a region of practical equivalence (ROPE) around zero, and obtain the posterior probability that the true value lies therein.
**plotACs**

**Value**
Returns an object of class histogram invisibly. Used mainly for the side effect.

**Author(s)**
Mike Meredith, adapted from code by John Kruschke.

**References**

**See Also**
postPlot.

**Examples**

```r
# See examples in dippers.
```

---

**plotACs**  
*Plot Activity Centres from Bsecr output*

**Description**
Plot posterior distributions of Activity Centre (AC) locations using the output from Bsecr.

**Usage**

```r
plotACs(object, which=NA, howMany=3000, showLabels=TRUE)
```

**Arguments**
- **object**: a Bwiqid object with ACs attribute
- **which**: a numeric vector indication which ACs to plot, default is to plot all.
- **howMany**: the number of points to plot for each AC
- **showLabels**: if TRUE, point clusters for animals detected will be labelled with the animal ID.

**Value**
Returns nothing, used for its plotting side effect.

**Author(s)**
Mike Meredith
Description

Plot the posterior probability distribution for a single parameter calculated using the comb method described by Kruschke (2015).

Usage

plotComb(x, y, credMass = 0.95, plot = TRUE, showMode = FALSE, shadeHDI = NULL, ...)

Arguments

x  A vector of equally-spaced possible values for the parameter. The range should cover all values of the parameter with non-negligible probability. (To restrict the range displayed in the plot, use xlim.)
y  A vector of probabilities corresponding to the values in x.
credMass  the probability mass to include in credible intervals; set to NULL to suppress plotting of credible intervals.
plot  logical: if TRUE, the posterior is plotted.
showMode  logical: if TRUE, the mode is displayed instead of the mean.
shadeHDI  specifies a colour to shade the area under the curve corresponding to the HDI; NULL for no shading. Use colours() to see a list of possible colours.
...  additional graphical parameters.

Details

The function calculates the Highest Density Interval (HDI). A multi-modal distribution may have a disjoint HDI, in which case the ends of each segment are calculated. No interpolation is done, and the end points correspond to values of the parameter in x; precision will be determined by the resolution of x.

If plot = TRUE, the probability density is plotted together with either the mean or the mode and the HDI.
plotComb

Value

Returns a matrix with the upper and lower limits of the HDI. If the HDI is disjoint, this matrix will have more than 1 row. It has attributes credMass and height, giving the height of the probability curve corresponding to the ends of the HDI.

Author(s)

Mike Meredith

See Also

For details of the HDI calculation, see hdi.

Examples

# Generate some data:
N <- 0:100
post <- dpois(N, 25)
# Do the plots:
plotComb(N, post)
plotComb(N, post, showMode=TRUE, shadeHDI='pink', xlim=c(10, 50))
# A bimodal distribution:
post2 <- (dnorm(N, 28, 8) + dnorm(N, 70, 11)) / 2
plotComb(N, post2, credMass=0.99, shade='pink')
plotComb(N, post2, credMass=0.80, shade='grey')

## S3 method for class 'wiqid'
predict(object, newdata, parameter, ci, type=c("link", "response"), ...)

### predict.wiqid

**Description**

Obtains predictions, with estimates, standard errors and confidence intervals, from a fitted model object of class *wiqid*, as produced by frequentist estimation functions in the *wiqid* package. Not all functions produce objects that enable predictions to be made; see Details. Please treat this as a 'beta' version and check output carefully.

### Usage

```r
## S3 method for class 'wiqid'
predict(object, newdata, parameter, ci, type=c("link", "response"), ...)
```

### Arguments

- **object**: an object of class *wiqid*.
- **newdata**: a data frame with columns for each of the covariates in the model. Unused columns are ignored. Missing values are not allowed. See Details.
- **parameter**: character; the name of the parameter to predict; this will appear on the left hand side of one of the formulae in the model.
- **ci**: the confidence interval to use; the default is to use `object$ci` or, if that is `NULL`, 0.95.
- **type**: the type of prediction required. The default is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable. Thus if the parameter is a probability, the default predictions are on the logit or probit scale and `type = "response"` gives the predicted probabilities. May be abbreviated.
- **...**: further arguments for other methods.

### Details

Most *wiqid* functions have models with multiple submodels, corresponding to the formulae in the model argument. Check `object$formulae` for a list of the available submodels.

The argument `newdata` is required (even for intercept-only models), and must be a data frame with named columns for each of the covariates in the submodel. For factors, the levels must be (a subset of) the levels in the original data; check `object$xlev` for possible levels.

`predict` is not yet implemented for the following functions:
predict.wiqid

occSSStime and occSScovSite: use occSS instead.
occMSStime and occMSCovSite: use occMS instead.
closedCap* functions: these models have no covariates.
surv* functions: these have no covariates.

Value

Returns a matrix with four columns (estimate, SE, lower and upper confidence limits) and a row for each row in newdata. If newdata has row names, these will be used for the output. Note that for an intercept-only submodel, all rows will have identical values. Attributes give information on the link used and the confidence level.

Author(s)

Mike Meredith.

Examples

# Generate some simulated occupancy data for 300 sites:
set.seed(2017)
original.data <- data.frame(
elev = runif(300, 0, 1000),
forType = factor(sample(c("dry", "swamp", "mangrove"), size=300, replace=TRUE, prob=3:1)))
modMat <- model.matrix(~ elev + forType, data = original.data)
psiCoef <- c(3, -0.003, -3, -1) # declines with elev; highest for 'dry', lowest 'mangrove'
psi <- plogis(modMat %*% psiCoef)
hist(psi, breaks=20)
z <- rbinom(300, 1, psi)
mean(z) # true realized occupancy
# detection history for 3 replicates, constant p = 0.6:
DH <- matrix(rbinom(300*3, 1, 0.6*z), nrow=300)
# fit models
m0 <- occSS(DH)
_mE <- occSS(DH, psi ~ elev, data = original.data)
m_EF <- occSS(DH, psi ~ elev + forType, data = original.data)

# now try predictions:
newdata <- expand.grid(elev=c(200, 500, 800), forType=c("dry", "swamp"))
predict(m_EF, newdata, "psi")

# do a nice plot
xx <- seq(0, 1000, length=51)
plotdata <- expand.grid(elev=xx, forType=c("dry", "swamp", "mangrove"))
toPlot <- predict(m_EF, plotdata, "psi", type='res')
plot(xx, rep(0.5, 51), type='n', las=1, ylim=range(toPlot),
    xlab="Elevation", ylab="Occupancy probability")
ciCols <- adjustcolor(c("lightgreen", 'skyblue', 'pink'), 0.5)
estCols <- c("darkgreen", 'blue', 'red')
for(i in 1:3) {

}
this1 <- toPlot[plotdata$forType == levels(plotdata$forType)[i], ]
polygon(c(xx, rev(xx)), c(this1[, 3], rev(this1[, 4])), col=ciCols[i])
lines(xx, this1[, 1], col=estCols[i])
}
legend('topright', levels(plotdata$forType), lty=1, col=estCols, bty='n')

# Add a survey-level covariate: observer ID with different detection probabilities
observer <- c(sample(1:2, size=300, replace=TRUE), # A and B on first survey occasion
 sample(1:3, size=300, replace=TRUE), # A, B and C for second
 sample(2:3, size=300, replace=TRUE)) # only B and C for third
obsID <- matrix(LETTERS[observer], nrow=300)
colnames(obsID) <- c("obs1", "obs2", "obs3")
original.data <- cbind(original.data, as.data.frame(obsID))
str(original.data)
p <- c(0.4, 0.6, 0.8)[observer]
DH <- matrix(rbinom(300*3, 1, p*z), nrow=300)
mEFO <- occSS(DH, list(psi ~ elev + forType, p ~ obs), data = original.data)

Description

Both functions print details of the MCMC process to the Console. print also prints a table of summary statistics for the parameters and several MCMC diagnostic measures to the Console, while summary returns invisibly a corresponding data frame, which can be passed to View.

Usage

## S3 method for class 'Bwiqid'
print(x, digits=3, ...)

## S3 method for class 'Bwiqid'
summary(object, digits=3, ...)

Arguments

x, object an object of class Bwiqid.
digits the number of digits to print or include in the output.
... further arguments for the print or summary function.
Details

The print function prints a table with a row for each parameter after removing duplicated rows. Duplication usually arises because a covariate has only a few unique values.

There are columns for each of the following summary statistics ... 

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>the mean of each MCMC chain.</td>
</tr>
<tr>
<td>sd</td>
<td>the standard deviation of each MCMC chain.</td>
</tr>
<tr>
<td>median</td>
<td>the median of each MCMC chain.</td>
</tr>
<tr>
<td>HDIlo and HDIup</td>
<td>the lower and upper values of a 95% Highest Density Interval CrI for each MCMC chain.</td>
</tr>
</tbody>
</table>

... and for some or all of the following diagnostics, depending on the MCMC engine used for fitting the model:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n.eff</td>
<td>the effective chain length for the parameters adjusted for autocorrelation; for stable estimates of credible intervals this should be at least 10,000. See <code>effectiveSize</code>.</td>
</tr>
<tr>
<td>MCerror</td>
<td>the Monte Carlo errors for the parameters, expressed as a percentage of the standard deviation. Values less than 5% are acceptable.</td>
</tr>
<tr>
<td>Rhat</td>
<td>the with potential scale reduction factors for the parameters, which is 1 on convergence and should be &lt; 1.05 for all.</td>
</tr>
</tbody>
</table>

Value

print returns x invisibly. summary returns the table of summary statistics.

Author(s)

Mike Meredith.

Examples

# See examples for dippers.

Priors

Standardisation and priors

Description

This page documents the priors used in the Bayesian analysis. For logistic models, sensible priors depend on the range of values and thus on the standardisation scheme used, which is also detailed here.

At present, this represents good intentions! It has not been implemented in all the functions in wiqid.

Standardisation

Continuous variables are standardised to facilitate writing models, and optimisation. Standardisation also means that the size of regression coefficients directly reflect the importance of the corresponding variables.
Binary variables coded as TRUE/FALSE and dummy variables are not changed. To make continuous variables comparable with these, they are centred by subtracting the mean, and then divided by their standard deviation.

Update: In versions prior to 0.2.x, continuous variables were centred by subtracting the mean, and then divided by two times their standard deviation (Gelman, 2008). With the new default, beta coefficients will be exactly the same size. There may be some rounding errors in the fourth decimal place for other estimates.

Note that all numerical inputs (ie, is.numeric == TRUE) that appear in the data argument will be standardised, including binary variables coded as 0/1. Variables coded as TRUE/FALSE or as factors are not affected.

The same standardisation strategy is used for both Bayesian and maximum likelihood functions.

Priors for logistic regression coefficients

Following Gelman et al (2008), we use independent Cauchy priors with centre 0 and scale 10 for the intercept and scale 2.5 for all other coefficients.

Priors for probabilities

We use independent Uniform(0, 1) priors for probabilities.

References


railSims

Simulated detection/non-detection data for two species of rails

Description

A data set for single-season two-species occupancy modelling. See occ2sps for details of these kinds of models.

Usage

data("railSims")

Format

A data frame with detection (1) vs non-detection data for 2 species at 160 sites on three occasions.

A1, A2, A3  detection histories for the dominant species on 3 occasions
B1, B2, B3  detection histories for the subordinate species on 3 occasions
logArea  a continuous site covariate, standardised to mean 0, sd 1
reeds  a logical site covariate.
**Details**

The data come from a simulated scenario with the following parameters:

- \( \psi_A = \text{plogis}(0 + 2 \times \text{logArea}) \) = probability of occupancy of species A
- \( \psi_{BA} = 0.77 \) = probability of occupancy of B if A is absent
- \( \psi_{BA} = \text{plogis}(-1 + 2 \times \text{reeds}) \) = probability of occupancy of B if A is present
- \( p_A = 0.75 \) = probability of detection of species A if B is absent
- \( r_A = p_A \) = probability of detection of species A if both are present
- \( p_B = 0.80 \) = probability of detection of species B if A is absent
- \( r_{Ba} = p_B \) = probability of detection of species B if both are present but A was not detected
- \( r_{BA} = 0.40 \) = probability of detection of species B if both are present and A was detected

**Source**

Simulated data

**References**


**Examples**

```r
data(railSims)
# Separate the two detection histories
DHA <- railSims[, 1:3]
DHB <- railSims[, 4:6]

# Default model (no interaction)
occ2sps(DHA, DHB)

# Model with full interaction
occ2sps(DHA, DHB, list(psiBA ~ 1, rA ~ 1, rBa ~ 1, rBA ~ 1))

# Model corresponding to the data generation model
occ2sps(DHA, DHB, list(psiA ~ logArea, psiBA ~ reeds, rBA ~ 1), data=railSims)
```

---

**richCurve**  
*Species richness estimates based on accumulation curves*

**Description**

Provides a shell into which species richness estimators may be plugged to provide estimates based on species accumulation curves, as provided by EstimateS.
Usage

richCurve(obsMat, FUNC, runs = 10, ...)

richSobs(incVec)
richSingle(cntVec)
richDouble(cntVec)
richUnique(incMat)
richDuplicate(incMat)

Arguments

incVec  a vector of species incidences (presences) in one or more samples; a vector of counts or a species x sites matrix of incidences or counts may be supplied.
cntVec  a vector of species counts (abundances); a species x sites matrix of counts may be supplied and will be converted to a vector with rowSums.
incMat  a 1/0 matrix of species incidence (presence), species x sites. A matrix of counts may also be provided.
obsmat  a matrix of species counts, species x sites; a matrix of incidences will be sufficient if accepted by FUNC.
FUNC   a function to estimate species richness based on a matrix of observations; see Species richness estimators for examples.
runs   the number of randomisations of samples (ie. columns in the input matrix) to perform to calculate mean and standard deviation.
...  additional arguments passed to FUNC.

Details

The reliability of estimates of species richness depends on the sampling effort. To investigate this effect, and judge whether the current sampling effort is adequate, we calculate richness estimates for subsets of the data. Assuming that the columns of the data matrix are independent samples from the population, richCurve calculates estimates for 1 sample, 2 samples, and so on. This is repeated for many runs, and the mean and standard deviation calculated.

The other functions documented here are trivial, but useful for plugging into richCurve:
richSobs : the number of species observed.
richSingle : the number of singletons, ie. species represented by just 1 individual in the pooled samples.
richDouble : the number of doubletons, ie. species represented by exactly 2 individuals in the pooled samples.
richUnique : the number of uniques, ie. species represented in just one sample.
richDuplicate : the number of duplicates, ie. species represented in exactly 2 samples.

Value

richCurve returns a list with elements:
A matrix (possibly 1-column) with a row for each sample and a column for each value returned by FUNC.

The corresponding matrix with the standard deviations of the estimates from the runs.

The other functions return scalars.

### Author(s)

Mike Meredith

### Examples

```r
data(seedbank)
plot(richCurve(seedbank, richSobs)$mean, type='l', ylim=c(0, 35))
lines(richCurve(seedbank, richSingle)$mean, col='blue')
lines(richCurve(seedbank, richDouble)$mean, col='blue', lty=2)
```

---

**richRarefy**

*Sample-based rarefaction curves*

### Description

Uses Mao’s tau estimator (Colwell et al, 2004) to obtain a rarefaction curve indicating the expected number of species observed if fewer samples were collected.

### Usage

```r
richRarefy(inpmat)
```

### Arguments

- **inpmat**: A 1/0 matrix of species incidence (presence), species x sites. A matrix of counts may also be provided.

### Value

A matrix with columns for the estimate and its standard deviation and rows for the number of samples pooled. Confidence limits may be obtained with estimate +/- 1.96 * SD.

### Author(s)

Mike Meredith

### References

Examples

data(seedbank)
plot(richRarefy(seedbank)[, 1], type='l')

Royle-Nichols occupancy model

Royle-Nichols model for single-season occupancy estimation

Description

These functions implement the Royle-Nichols method (Royle & Nichols 2003) for estimation of site occupancy allowing for abundance-induced heterogeneity in detection probability. Probability of detection is modelled as a function of the number of animals available for detection, \( n \), and the probability of detection of an individual animal, \( r \). Probability of occupancy is derived as the probability that \( n > 0 \).

Function \texttt{occSSrn} allows for site-specific covariates to be included in the model. \texttt{occSSrnSite} and \texttt{occSSrn0} are fast alternatives that do not require a full detection history matrix.

Usage

\[
\texttt{occSSrn(DH, model=NULL, data = NULL, ci=0.95, link=c("logit", "probit"), verify=TRUE, ...)}
\]

\[
\texttt{occSSrn0(y, n, ci=0.95, link=c("logit", "probit"), ...)}
\]

\[
\texttt{occSSrnSite(y, n, model=NULL, data = NULL, ci=0.95, link=c("logit", "probit"), ...)}
\]

Arguments

\texttt{DH}  
a 1/0/NA matrix (or data frame) of detection histories, sites x occasions.

\texttt{model}  
a list of formulae symbolically defining a linear predictor for each parameter in terms of covariates. If NULL, an intercept-only model is used, ie, \texttt{lambda(.) r(.)}.

\texttt{data}  
a data frame containing the variables in the model, with a row for each site. Each site covariate has one column. Each survey covariate has one column for each occasion, and the column name must end with the occasion number (without leading zeros); eg, \texttt{Cov1, Cov2, \ldots, Cov15}. All covariates should be included in \texttt{data}, otherwise they will be sought in enclosing environments, which may not produce what you want – and they won’t be standardised.

Note: currently only site covariates can be handled.

\texttt{ci}  
the confidence interval to use.

\texttt{link}  
the link function to use, either logit or probit; see \texttt{Links}.

\texttt{verify}  
if TRUE, the data provided will be checked.
Royle-Nichols occupancy model

y a vector with the number of detections at each site.
n a scalar or vector with the number of visits (survey occasions) at each site.

Details

Numeric covariates in data are standardised to facilitate convergence. This applies to binary covariates coded as 1/0; if this is not what you want, code these as TRUE/FALSE or as factors.

Value

Returns an object of class `wiqid`, see `wiqid-class` for details.

Benchmarks

Output has been checked against output from PRESENCE (Hines 2006) v.6.9 for the `weta` data set. Real values are mostly the same to 4 decimal places, though there is occasionally a discrepancy of 0.001. AICs are the same.

Author(s)

Mike Meredith

References


See Also

See the examples for the `weta` data set. See `occ2sps` for single-season two-species models and `occMS` for multi-season models.

Examples

# The weta data from MacKenzie et al (2006) p116:
data(weta)
DH <- weta[, 1:5]
occSS(DH) # for comparison
occSSrn(DH)
y <- rowSums(DH, na.rm=TRUE)
n <- rowSums(!is.na(DH))
occSSrnSite(y, n, lambda ~ Browsed, data=weta)
salamanders

Occupy data for blue ridge salamanders

Description
Detection/non-detection data for blue ridge salamanders (Eurycea wilderae) in Great Smoky Mountains National Park.

Usage
data(salamanders)

Format
A matrix with 39 rows corresponding to sites and 5 columns corresponding to survey occasions. 1 means that one or more salamanders were observed at the site/survey, 0 means none were seen.

Source
Described in MacKenzie et al (2006) p99. The data are distributed with the software package PRESENCE.

References

Examples
data(salamanders)
occSStime(salamanders, p ~ .time)

secrFit
Spatially Explicit Capture-Recapture

Description
A wrapper for secr::secr.fit. In secr v. 4, secr.fit gains a new option, fastproximity. If TRUE, some data sets are compressed and reconfigured to run much faster. This cannot be implemented for all models. The default is fastproximity=TRUE. This means that you can have a set of models where some have been reconfigured, others not, and AICs are not comparable across these models. The function secrFit simply calls secr.fit with fastproximity = FALSE, making it easy to run models with consistent settings.
seedbank

Usage

```r
secrFit(capthist, model = list(D~1, g0~1, sigma~1), mask = NULL, buffer = NULL,
        CL = FALSE, detectfn = NULL, ...)
```

Arguments

- `capthist`: a `capthist` object as defined in package `secr` including capture data and detector (trap) layout
- `model`: list with optional components each symbolically defining a linear predictor for one real parameter using formula notation
- `mask`: a mask object or (for a multi-session analysis) a list of mask objects, one for each session
- `buffer`: scalar mask buffer radius if mask is not specified (default 100 m)
- `CL`: logical, if true then the model is fitted by maximizing the conditional likelihood
- `detectfn`: integer code or character string for shape of detection function: `0` = halfnormal, `1` = hazard rate etc. - see `detectfn`
- `...`: other arguments to pass to `secr.fit`.

Value

returns an object of class `secr` representing the fitted SECR model.

Author(s)

This wrapper by Mike Meredith

Description

Number of seeds of different species germinating from 121 soil samples collected in a tropical secondary forest.

Usage

```r
data(seedbank)
```

Format

A matrix with 34 rows for species and 121 columns corresponding to different soil samples.

Source

Butler & Chazdon 1998. Example data distributed with the EstimateS program (Colwell 2005).
showShinyApp

References


Examples

data(seedbank)
##

| showShinyApp | Display a 'shiny' application |

Description

Displays one of the built-in interactive 'shiny' applications in the browser. See Details for the apps available.

Usage

showShinyApp(topic)

Arguments

topic The name of the shiny app to display. If missing, a list of available apps will be returned. Partial matching can be used.

Details

Three apps are currently included in the *wiqid* package:

"Beta" displays a beta distribution and sliders which allow you to change the parameters. You can also input binomial data and obtain the conjugate beta posterior distribution.

"Gamma" displays a gamma distribution with variable parameters, and can produce the conjugate gamma posterior for Poisson-distributed count data.

"Quadratic" plots a quadratic relationship with variable parameters, showing how the quadratic term can add a hump or hollow to a relationship.

Value

If topic is missing, a list of available apps. Otherwise, nothing useful; the function is run for its side effect.
simpleRhat

Author(s)
A much simplified version of code by Jason Bryer on GitHub at https://github.com/jbryer/IS606, adapted by Mike Meredith.

Examples
showShinyApp() # Shows a list of available apps

simpleRhat object n.chains burnin=0

Arguments
object an object of any class with MCMC output that can be coerced to class mcmcOutput.
n.chains ignored
burnin ignored

Value
A named vector with the Rhat values.

Author(s)
Mike Meredith

References

Species richness estimators

Examples

```r
# Get some output to use
data(salamanders)
y <- rowSums(salamanders)
(out <- BoccSS0(y, 5))
simpleRhat(out)
```

Species richness estimators

Description

Functions to estimate species richness, based on samples from one or more surveys (quadrats, sites, occasions, ...) as included in EstimateS. See Details for individual estimators.

Usage

```r
richACE(cntVec, threshold = 10)
richICE(incMat, threshold = 10)
richChao1(cntVec, correct = FALSE, ci = 0.95)
richChao2(incMat, correct = FALSE, ci = 0.95)
richJack1(incMat)
richJack2(incMat)
richJackA1(cntVec)
richJackA2(cntVec)
richBoot(incMat)
richMM(incMat)
richRenLau(cntVec)
```

Arguments

- `cntVec` - a vector of species counts (abundances) with one element for each species. A matrix of counts, species x sites, may also be provided and will be converted to a vector with `rowSums`. Zeros are allowed, but not missing values; non-integers are rounded.
- `incMat` - a 1/0 matrix of species incidence (presence), species x sites. A matrix of counts may also be provided and will be converted to 1/0 after rounding.
- `threshold` - the definition of rare or infrequent species: species with `threshold` or smaller counts (ACE) or incidences (ICE) are rare or infrequent.
- `correct` - if TRUE, bias-corrected estimates are calculated.
- `ci` - the required confidence interval.
Species richness estimators

Details

richACE and richICE calculate Anne Chao’s Abundance-based and Incidence-based Coverage Estimators of species richness respectively (Chao et al, 2000).

richChao1 and richChao2 calculate Anne Chao’s Chao 1 (abundance-based) and Chao 2 (incidence-based) estimators (Chao 1984, 1987).

richJack1 and richJack2 calculate first and second order incidence-based jackknife estimators of species richness (Smith & van Belle, 1984).

richBoot calculates a bootstrap estimator of species richness (Smith & van Belle, 1984).

richMM calculates the asymptotic species richness from a Michaelis-Menten curve fitted to the species rarefaction curve (Colwell et al. 2004).

See the EstimateS Users Guide at http://viceroy.eeb.uconn.edu/estimates/ for the equations for the functions above.

The following are not included in EstimateS v.8.2:

richJackA1 and richJackA2 calculate first and second order abundance-based jackknife estimators of species richness (Gotelli & Colwell 2011).


Value

richChao1 and richChao2 return a vector with a point estimate, upper and lower confidence limits, and standard deviation.

The other functions return a scalar.

Benchmarks

Output for estimators included in EstimateS 8.2 has been checked against EstimateS for the seedbank and killarneyBirds data sets. EstimateS results are often 0.01 lower, as EstimateS appears to truncate rather than rounding.

Author(s)

Mike Meredith

References


See Also

richRarefy for rarefaction curves, and richCurve for a function to give richness estimates for sub-sets of samples.

Examples

data(seedbank)

richACE(seedbank)

standardize Scaling and centring of vectors, matrices and arrays

Description

Maps a numeric variable to a new object with the same dimensions. standardize is typically used to standardise a covariate to mean 0 and SD 1. standardize2match is used to standardise one object using the mean and SD of another; it is a wrapper for standardize(x,center=mean(y),scale=sd(y)).

Usage

standardize(x, center = TRUE, scale = TRUE)
standardize2match(x, y)

Arguments

x, y a numeric vector, matrix or multidimensional array; NA and NaN are allowed; Inf and -Inf will produce all-NaN output if either center or scale is TRUE.

center either a logical or a numeric value of length 1.

scale either a logical or a numeric value of length 1.
Details

`standardize` differs from `scale` by (1) accepting multidimensional arrays but not data frames; (2) not standardizing column-wise but using a single value to centre or to scale; (3) if `x` is a vector, the output will be a vector (not a 1-column matrix). If each column in the matrix represents a different variable, use `scale` not `standardize`.

Centring is performed before scaling.

If `center` is numeric, that value will be subtracted from the whole object. If logical and TRUE, the mean of the object (after removing NAs) will be subtracted.

If `scale` is numeric, the whole object will be divided by that value. If logical and TRUE, the standard deviation of the object (after removing NAs) will be used; this may not make sense if `center` = FALSE.

Value

A numeric object of the same dimensions as `x` with the standardized values. NAs in the input will be preserved in the output.

For the default arguments, the object returned will have mean approximately zero and SD 1. (The mean is not exactly zero as scaling is performed after centring.)

Author(s)

Mike Meredith, after looking at the code of `base::scale`.

Examples

```r
# Generate some fake elevation data:
elev <- runif(100, min=100, max=500)
mean(elev) ; sd(elev)
str( e <- standardize(elev) )
mean(e) ; sd(e)

# Standardize so that e=0 corresponds to exactly 300m and +/- 1 to a change of 100m:
e <- standardize(elev, center=300, scale=100)
mean(e)
mean(elev) - 300
range(e)
range(elev) - 300

# Generate data matrix for survey duration for 3 surveys at 10 sites
dur <- matrix(round(runif(30, 20, 60)), nrow=10, ncol=3)
d <- standardize(dur)
mean(d) ; sd(d)

# Standardize new data to match the mean and SD of 'dur'
(new <- seq(20, 60, length.out=11))
standardize2match(new, dur)

# compare with base::scale
```
dx <- base::scale(dur)
colMeans(dx) ; apply(dx, 2, sd)
colMeans(d) ; apply(d, 2, sd)
# Don't use 'standardize' if the columns in the matrix are different variables!

Survival from recapture data with Cormack-Jolly-Seber (CJS) model

**Description**

Calculation of apparent survival (accounting for recapture probability) from mark-recapture data, with time-dependent phi or p, possibly with covariates. Function `survCHSaj` allows for different survival parameters for juveniles and adults; juveniles are assumed to become adults after the first interval. `BsurvCJS` is a Bayesian version.

**Usage**

```r
survCJS(DH, model=list(phi~1, p~1), data=NULL, freq=1, group, interval=1,
        ci = 0.95, link=c("logit", "probit"), ...)

survCJSaj(DHj, DHa=NULL, model=list(phiJ~1, phiA~1, p~1), data=NULL,
         freqj=1, freqa=1, ci = 0.95, link=c("logit", "probit"), ...)

BsurvCJS(DH, model=list(phi~1, p~1), data = NULL, freq=1, priors=NULL,
         chains=3, draws=1e4, burnin=1000, thin=1, adapt=1000,
         parallel = NULL, seed=NULL, priorOnly=FALSE, ...)
```

**Arguments**

- **DH**
  a 1/0 matrix with detection histories with a row for each animal captured and a column for each capture occasion.

- **model**
  a list of formulae symbolically defining a linear predictor for each parameter in terms of covariates.

- **data**
  a data frame with a row for each survival interval / recapture occasion and columns for each of the covariates used to estimate phi or p.

- **freq**
  a scalar or a vector of length `nrows(DH)` with the frequency of each detection history. Negative values indicate trap losses.

- **group**
  an optional factor of length `nrows(DH)`; if provided, group can be included in the model definition, see Examples.

- **interval**
  the time interval between capture occasions; scalar if all intervals are equal or a vector of length `ncole(DH) - 1`; the units used must be the same as those for the apparent survival estimate, eg, for annual survival use years.

- **DHj, DHa**
  detection history matrices for animals marked as juveniles and adults respectively; `DHa` should be `NULL` if no animals were marked as adults.

- **freqj, freqa**
  frequencies of each detection history in `DHj` and `DHa`; `freqa` is ignored if `DHa = NULL`. 

---

Survival (CJS)
ci the required confidence interval.
link the link function to use, either logit or probit; see Links.
... other arguments passed to nlm.
priors a list with elements for prior mean and variance for coefficients; see Details.
chains the number of Markov chains to run.
draws the minimum number of values to return; the actual number returned may be
slightly higher, as it will be a multiple of chains.
burnin the number of values to discard at the beginning of each chain.
thin the thinning rate. If set to n > 1, n values are calculated for each value returned.
adapt the number of iterations to run in the JAGS adaptive phase.
priorOnly if TRUE, the function produces random draws from the appropriate prior distri-
butions, with a warning.
parallel if TRUE or NULL and sufficient cores are available, the MCMC chains are run
in parallel; if TRUE and insufficient cores are available, a warning is given.
seed a positive integer, the seed for the random number generators.

Details

BsurvCJS uses a probit link to model apparent survival and detection as a function of covariates;
most software uses a logistic (logit) link. See Links. Coefficients on the probit scale are about half
the size of the equivalent on the logit scale.

Priors for BsurvCJS are listed in the priors argument, which may contain elements:

muPhi and muP : the means for apparent survival and detection coefficients respectively. This may
be a vector with one value for each coefficient, including the intercept, or a scalar, which will be
used for all. The default is 0.
sigmaPhi and sigmaP : the variance for apparent survival and detection coefficients respectively.
This may be (1) a vector with one value for each coefficient, including the intercept, which repre-
sents the variance, assuming independence, or (2) a scalar, which will be used for all. The function
does not currently allow a variance-covariance matrix. The default is 1, which is somewhat infor-
mative.

When specifying priors, note that numerical covariates are standardized internally before fitting the
model. For an intercept-only model, a prior of Normal(0, 1) on the probit scale implies a Uniform(0,
1) or Beta(1, 1) prior on the probability scale.

Value

survCJS and survCJSaj return an object of class wiqid, a list with elements:
call The call used to produce the results
beta Estimates of the coefficients in the linear predictors for phi and p.
beta.vcv The variance-covariance matrix for the beta estimates.
real Back-transformed estimates of phi and p for each interval / occasion.
logLik  

a vector with elements for log(likelihood), number of parameters, and effective sample size. If the variance-covariance matrix cannot be calculated, the second element should be NA.

There are print, logLik, and nobs methods for class wiqid.

BsurvCJS returns an object of class Bwiqid, a data frame with columns for each p and psi value containing the series of MCMC draws, and attributes for details of the MCMC run.

Benchmarks

Output of survCJS has been checked against program MARK with the dipper data set: coefficients are not the same as MARK uses models without an intercept, but the real values agree to 3 decimal places.

Author(s)

Mike Meredith

References


Examples

data(dippers)

DH <- dippers[1:7]  # Extract the detection histories
survCJS(DH)  # the phi(.) p(.) model
survCJS(DH, phi ~ .time)  # the phi(t) p(.) model
df <- data.frame(flood = c(FALSE, TRUE, TRUE, FALSE, FALSE, FALSE))
survCJS(DH, phi ~ flood, data=df)  # the phi(flood) p(.) model
# Including a grouping factor:
survCJS(DH, phi ~ flood*group, data=df, group=dippers$sex)

# With unequal intervals - suppose no data were collected in year 5:
DH1 <- DH[, -5]
survCJS(DH1, phi ~ .time, interval = c(1, 1, 1, 2, 1))

# See also the examples in the dippers help file.
Description

Calculation of apparent survival and recruitment rate from data collected using Pollock’s robust design, i.e., with multiple capture occasions within each season, where the population is closed within each season.

Function `survRDah` implements the second stage of a two-stage analysis, where abundance and recapture probability are estimated using closed-capture function for each season.

Function `survRD` combines the two stages into a single maximum likelihood estimation step, using model M0 for the within-season data.

NOTE: These are preliminary attempts at coding these models and have not been properly tested or benchmarked.

Usage

```r
survRD(DH, freq=1, occsPerSeason)
survRDah(DH, freq=1, occsPerSeason, N, pStar)
```

Arguments

- **DH**: a 1/0 matrix with detection histories with a row for each animal captured and a column for each capture occasion.
- **freq**: a scalar or a vector of length `nrow(DH)` with the frequency of each detection history. Negative values indicate trap losses.
- **occsPerSeason**: the number of survey occasions per season; currently this must be scalar and the number of occasions must be the same for all seasons.
- **N**: a vector with an element for each season giving the number of animals available for capture as estimated in the first stage of a 2-stage analysis.
- **pStar**: a vector with an element for each season giving the probability of recapture as estimated in the first stage of a 2-stage analysis.

Value

A list with elements:

- **phiHat**: Estimates of apparent survival for each interval between seasons.
- **bHat**: Estimates of the recruitment rate for each interval.
- **pStarHat**: The estimated probability of capture during each season.
- **Nhat**: The estimated number of animals available for capture during each season.
- **pHat**: The estimated probability of capture on one occasion for each season.

For `survRDah`, the values of `pStarHat` and `Nhat` will equal the values of `pStar` and `N` input, and `pHat` will be NULL.

Author(s)

Mike Meredith
References


Examples

data(MeadowVoles)

# Extract detection histories:
DH <- MeadowVoles[, 1:30]
freq <- MeadowVoles$freq

# With single stage maximum likelihood estimation:
survRD(DH, freq=freq, occsPerSeason=5)

# The 2-stage approach:
# Stage 1 - using the jackknife estimator to estimate N and p for each season:
MhResult <- matrix(NA, 6, 2)
colnames(MhResult) <- c("N", "p")
seasonID <- rep(1:6, each=5)
for(i in 1:6) {
  dh <- DH[, seasonID==i]
  MhResult[i, ] <- closedCapMhJK(dh)$real[, 1]
}
MhResult

# Calculate the probability of being captured at least once in the season:
pStar <- 1 - (1 - MhResult[, "p"]^5)

# Stage 2 - pass N and pStar to a modified CJS estimation routine:
survRDah(DH, freq=freq, occsPerSeason=5, N=MhResult[, "N"], pStar=pStar)


TDist

The Generalized Student’s $t$ Distribution

Description

Density, distribution function, quantile function and random generation for the generalised $t$ distribution with df degrees of freedom, using location and scale, or mean and sd. These are wrappers for stats::dt, etc.

Note: In earlier versions of wiqid the scale argument to *t2* functions was incorrectly named sd; they are not the same. These function now give a warning with the correct value of the sd. New *t3* functions do use sd which is only defined for df > 2.
Usage

\begin{verbatim}
dt2(x, location, scale, df)
pt2(x, location, scale, df, lower.tail=TRUE, log.p=FALSE)
qt2(p, location, scale, df, lower.tail=TRUE, log.p=FALSE)
rt2(n, location, scale, df)
dt3(x, mean, sd, df)
pt3(x, mean, sd, df, lower.tail=TRUE, log.p=FALSE)
qt3(p, mean, sd, df, lower.tail=TRUE, log.p=FALSE)
rt3(n, mean, sd, df)
\end{verbatim}

Arguments

- **x**: vector of parameter values
- **location**: location of the t-distribution
- **mean**: mean of the t-distribution
- **scale**: scale parameter of the t-distribution
- **sd**: standard deviation of the t-distribution, only defined for \( df > 2 \).
- **df**: degrees of freedom
- **lower.tail**: logical; if TRUE (default), cumulative probabilities up to \( x \), otherwise, above \( x \).
- **log.p**: logical; if TRUE, probabilities \( p \) are given as \( \log(p) \).
- **p**: probability.
- **n**: number of random draws required.

Value

\( dtx \) gives the density, \( ptx \) gives the cumulative probability, \( qtx \) gives the quantile function, and \( rtx \) generates random deviates.

Author(s)

Mike Meredith

See Also

See the \texttt{stats} functions \texttt{dt}, \texttt{pt}, \texttt{qt}, \texttt{rt}.

Examples

\begin{verbatim}
## Plot the t-distribution with varying sd and scale
require(graphics)
xx <- seq(-5, 15, length=201)
density <- dt3(xx, mean=5, sd=1, 4)
plot(xx, density, type=\'l\', lwd=2, main="t-distribution with df = 4")
lines(xx, dt3(xx, 5, 2, 4), lwd=2, col=2)
lines(xx, dt3(xx, 5, 3, 4), lwd=2, col=3)
lines(xx, dt3(xx, 5, 4, 4), lwd=2, col=4)
\end{verbatim}
Temburong

```
legend('topleft', paste0("sd = ", 1:4), lwd=2, lty=1, col=1:4, bty='n')
lines(xx, dt2(xx, 5, 1, 4), lwd=2, lty=2, col=1)
lines(xx, dt2(xx, 5, 2, 4), lwd=2, lty=2, col=2)
lines(xx, dt2(xx, 5, 3, 4), lwd=2, lty=2, col=3)
lines(xx, dt2(xx, 5, 4, 4), lwd=2, lty=2, col=4)
legend('topright', paste0("scale = ", 1:4), lwd=2, lty=2, col=1:4, bty='n')
```

```
# Generate random numbers
rand2 <- rt2(1e6, location=5, scale=2, df=4)
mean(rand2)
sd(rand2)  # about 2.83
rand3 <- rt3(1e6, mean=5, sd=2, df=4)
mean(rand3)
sd(rand3)  # close to 2
```

```
# check pt* and qt*
prob <- pt2(x=7, location=5, scale=3, df=4)
qt2(p=prob, location=5, scale=3, df=4)
```

---

**Temburong**

**Tree species count data**

**Description**

Basal area and number of individual trees >= 5cm dbh of each species in a 1ha plot in Ulu Temburong - Brunei. There were 1012 trees of 276 species.

**Usage**

data(Temburong)

**Format**

The data set consists of two objects:

- Temburong is a vector of length 276, with the counts of each species.
- TemburongBA is a similar vector with the basal area (ie. the sum of the cross-sectional area at breast height of all of trees) of each species.

**Source**


**Examples**

data(Temburong)
richChao1(Temburong)
biodShannon(TemburongBA)
**WAIC**

Extract the Watanabe-Akaike Information Criterion (WAIC)

---

**Description**

Extracts the Watanabe-Akaike Information Criterion from objects of class `Bwiqid` which have a WAIC attribute.

**Usage**

```r
WAIC(object, ...)  
```

**Arguments**

- `object`  
  a fitted model object with an attribute giving the WAIC and pD for the fitted model.

- `...`  
  optionally more fitted model objects.

**Value**

- If just one object is provided, the corresponding WAIC.
- If multiple objects are provided, a data frame with rows corresponding to the objects and columns representing the number of parameters in the model (pD) and the WAIC.

**Note**

The code to produce WAIC is new and not fully tested: it probably harbours bugs!

**Author(s)**

Mike Meredith.

**References**


**Examples**

```r
## TODO
```
Detection data for weta in gorse bushes

Description

Results of an occupancy survey to see if presence/absence of weta in 72 gorse bushes is affected by browsing of the bushes by goats. Probability of detection may be different for each observer, and observer ID is also recorded.

Usage
data(weta)

Format

A data frame with 72 observations on the following 11 variables.

**D1, D2, D3, D4, D5**  
a numeric vector for each of 5 daily surveys, showing whether weta were detected (1) or not detected (0); NA if the bush was not surveyed on the relevant day.

**Browsed**  
a logical vector indicating whether the bush was browsed (TRUE) or not browsed (FALSE)

**ObsD1, ObsD2, ObsD3, ObsD4, ObsD5**  
a factor with levels A B C, indicating which observer carried out each survey.

Details

The data are provided as a data frame, such as would result from reading in data from a text file. Further formatting is needed before using these for analysis: see the examples.

Source


References


Examples

data(weta)

```
DH <- weta[, 1:5]  # extract the detection history:
occSS(DH)
occSStime(DH, p~.time)
# With covariates
occs(time=DH, list(psi ~ Browsed, p ~ ObsD), data=weta)
occs(time=DH, list(psi ~ Browsed, p ~ Browsed), data=weta)
```
# Bayesian analysis

# Model with separate intercepts for occupancy in Browsed and Unbrowsed
# bushes, and a time trend for probability of detection; specify uniform
# priors for probability of occupancy:
Bwet <- BoccSS(DH, model=list(psi~Browsed-1, p~.Time), data=weta,
               priors=list(sigmaPsi=c(1,1)), chains=2)
Bwet
plot(Bwet)
plot(Bwet, "p_.Time")

---

**wiqid-class**  
*The 'wiqid' S3 class*

**Description**

All the maximum likelihood functions in the **wiqid** package should produce an object of class **wiqid**. Bayesian estimation runs produce objects of class **Bwiqid**.

**Structure**

A **wiqid** object is a list with the following elements:

- `call`  The call used to produce the results
- `link`  Either a named list with the link functions used, or a single value if the same link is used for all parameters. For probabilities, usually `logit` or `probit`.
- `beta`  A matrix of values of the coefficients of the terms in the linear predictors, with standard errors and confidence intervals.
- `beta.vcv`  The variance-covariance matrix for the beta estimates.
- `real`  Estimates of occupancy and probability of detection on the real scale, with confidence intervals.
- `logLik`  A vector with elements for `log(likelihood)`, number of parameters, and effective sample size. If the variance-covariance matrix cannot be calculated, the second element should be `NA`.
- `ci`  Intended coverage of the confidence intervals.
- `formulae`  A named list with the formulae of each of the submodels.
- `index`  A named list indicating which rows of the beta and beta.vcv matrices correspond to each submodel.
- `xlev`  A named list of factors included in the original data object with their levels.
- `scaling`  A named list of numerical covariates included in the original data object with the values used to standardise them, `c(centre, spread)`.

**Methods**

The following methods are available for objects of class **wiqid**:
- `print`, `logLik`, `nobs`, `coef`, `vcov`, `predict.wiqid`. 
Index

*Topic datasets
  dippers, 26
  distTestData, 31
  GrandSkinks, 36
  KanhaTigers, 37
  KillarneyBirds, 38
  MeadowVoles, 40
  railSims, 55
  salamanders, 61
  seedbank, 62
  Temburong, 75
  weta, 77
*Topic hplot
  plot.Bwiqid, 47
  plotComb, 49
*Topic manip
  Links, 39
  Priors, 54
*Topic methods
  Bwiqid-class, 22
  plot.Bwiqid, 47
  predict.wiqid, 51
  print.Bwiqid, 53
*Topic models
  Links, 39
  Priors, 54
  wiqid-class, 78
*Topic package
  wiqid-package, 3
*Topic predict
  predict.wiqid, 51
*Topic print
  print.Bwiqid, 53

AIC, 8
AICc, 6, 7
AICtable, 6, 9
allCombinations, 6, 10
as.Bwiqid (Bwiqid-class), 22

Bayesian binomial analysis, 11
Bayesian normal estimation, 12
Bayesian Occupancy Single Season, 14
Bayesian Poisson analysis, 17
Bayesian SCR, 18
Bbinom (Bayesian binomial analysis), 11
Bbinomial, 3
Bbinomial (Bayesian binomial analysis), 11
BetaDist, 19
biodBerger, 5
biodBerger (Diversity indices), 32
biodBrillouin, 5
biodBrillouin (Diversity indices), 32
biodShannon, 5
biodShannon (Diversity indices), 32
biodSimpson, 5
biodSimpson (Diversity indices), 32
Bnormal, 3
Bnormal (Bayesian normal estimation), 12
Bnormal2 (Bayesian normal estimation), 12
BoccSS, 4
BoccSS (Bayesian Occupancy Single Season), 14
BoccSS0, 4
BoccSS0 (Bayesian Occupancy Single Season), 14
Bpoisson, 3
Bpoisson (Bayesian Poisson analysis), 17
Bsecr0, 4, 48
Bsecr0 (Bayesian SCR), 18
BsurvCJS, 4
BsurvCJS (Survival (CJS)), 69
Bwiqid-class, 22

Closed Captures, 22
closedCapM0, 4
closedCapM0 (Closed Captures), 22
closedCapMb, 4

79
INDEX

closedCapMb (Closed Captures), 22
closedCapMh2, 4
closedCapMh2 (Closed Captures), 22
closedCapMhJK, 4
closedCapMhJK (Closed Captures), 22
closedCapMt, 4
closedCapMt (Closed Captures), 22
closedCapMtcov, 4
closedCapMtcov (Closed Captures), 22
coeff, 78
dbeta, 20
dbeta2, 6
dbeta2 (BetaDist), 19
dbeta3, 6
dbeta3 (BetaDist), 19
deprecated, 25
detectfn, 62
dgamma, 35
dgamma2, 6
dgamma2 (GammaDist), 34
dippers, 6, 26
dist, 29–31
Distance Measures, 27, 30, 31
distBrayCurtis, 5
distBrayCurtis (Distance Measures), 27
distChaoJaccCorr, 6
distChaoJaccCorr (Distance Measures), 27
distChaoJaccNaive, 6
distChaoJaccNaive (Distance Measures), 27
distChaoSorCorr, 6
distChaoSorCorr (Distance Measures), 27
distChaoSorNaive, 6
distChaoSorNaive (Distance Measures), 27
distChord, 6
distChord (Distance Measures), 27
distJaccard, 6
distJaccard (Distance Measures), 27
distMatching (Distance Measures), 27
distMorisitaHorn, 6
distMorisitaHorn (Distance Measures), 27
distOchiai, 6
distOchiai (Distance Measures), 27
distPreston, 6
distPreston (Distance Measures), 27
distRogersTanimoto, 6
distRogersTanimoto (Distance Measures), 27
distShell, 5, 27, 30
distSimRatio, 6
distSimRatio (Distance Measures), 27
distSorensen, 6
distSorensen (Distance Measures), 27
distTestData, 6, 31
distWhittaker, 6
distWhittaker (Distance Measures), 27
Diversity indices, 32
dt, 74
dt2, 6
dt2 (TDist), 73
dt3, 6
dt3 (TDist), 73
effectiveSize, 54
GammaDist, 34
gelman.diag, 54
getBeta2Par (BetaDist), 19
getBeta3Par (BetaDist), 19
getGammaPar (GammaDist), 34
getMCE, 35
getMCerror, 35
getRhat, 64
GrandSkinks, 6, 36, 44
hd, 50
KanhaTigers, 6, 37
KillarneyBirds, 6, 38
Links, 15, 39, 45, 59, 70
links (Links), 39
logLik, 7, 78
MeadowVoles, 6, 40
nlm, 23, 43, 45, 60, 70
nobs, 78
occ2sps, 4, 41, 46, 55, 60
occMS, 4, 46, 60
occMS (Occupancy Multi-Season), 42
occMS0, 4
occMS0 (Occupancy Multi-Season), 42
occMScovSite, 4
occMScovSite (Occupancy Multi-Season), 42
occMStime, 4
INDEX

occMStime (Occupancy Multi-Season), 42
occSS, 4, 42
occSS (Occupancy Single Season), 44
occSS0, 4
occSS0 (Occupancy Single Season), 44
occSScovSite, 4
occSScovSite (Occupancy Single Season), 44
occSSrn, 4
occSSrn (Royle-Nichols occupancy model), 59
occSSrn0 (Royle-Nichols occupancy model), 59
occSSrnSite (Royle-Nichols occupancy model), 59
occSSStime, 4
occSSStime (Occupancy Single Season), 44
Occupancy Multi-Season, 42
Occupancy Single Season, 44

pbeta, 20
pbeta2 (BetaDist), 19
pbeta3 (BetaDist), 19
pgamma, 35
pgamma2 (GammaDist), 34
plot.Bwiqid, 47
plotACs, 48
plotComb, 49
plotPost (deprecated), 25
postPlot, 48
predict.wiqid, 51, 78
print.Bwiqid, 53
Priors, 54
pt, 74
pt2 (TDist), 73
pt3 (TDist), 73

qbeta, 20
qbeta2 (BetaDist), 19
qbeta3 (BetaDist), 19
qgamma, 35
qgamma2 (GammaDist), 34
qt, 74
qt2 (TDist), 73
qt3 (TDist), 73

railSims, 6, 41, 42, 55
rbeta, 20
rbeta2 (BetaDist), 19

rbeta3 (BetaDist), 19
rgamma, 35
rgamma2 (GammaDist), 34
richACE, 5
richACE (Species richness estimators), 65
richBoot, 5
richBoot (Species richness estimators), 65
richChao1, 5
richChao1 (Species richness estimators), 65
richChao2, 5
richChao2 (Species richness estimators), 65
richCurve, 5, 56, 67
richDouble, 5
richDouble (richCurve), 56
richDuplicate, 5
richDuplicate (richCurve), 56
richICE, 5
richICE (Species richness estimators), 65
richJack1, 5
richJack1 (Species richness estimators), 65
richJack2, 5
richJack2 (Species richness estimators), 65
richJackA1, 5
richJackA1 (Species richness estimators), 65
richJackA2, 5
richJackA2 (Species richness estimators), 65
richMM, 5
richMM (Species richness estimators), 65
richRarefy, 5, 58, 67
richRenLau, 5
richRenLau (Species richness estimators), 65
richSingle, 5
richSingle (richCurve), 56
richSobs, 5, 34
richSobs (richCurve), 56
richUnique, 5
richUnique (richCurve), 56
Royle-Nichols occupancy model, 59
rt, 74
rt2 (TDist), 73
rt3 (TDist), 73

salamanders, 6, 46, 61
scale, 6, 68
secre.fit, 61
secrFit, 61
seedbank, 6, 62
showShinyApp, 63
simpleRhat, 64
Species richness estimators, 34, 57, 65
Standardisation (Priors), 54
Standardization (Priors), 54
standardize, 6, 67
standardize2match (standardize), 67
summary.Bwiqid (print.Bwiqid), 53
survCJS, 4
survCJS (Survival (CJS)), 69
survCJSaj, 4
survCJSaj (Survival (CJS)), 69
Survival (CJS), 69
Survival (RD), 71
survRD, 5
survRD (Survival (RD)), 71
survRDah, 5
survRDah (Survival (RD)), 71

TDist, 73
Temburong, 6, 75
TemburongBA, 6
TemburongBA (Temburong), 75

vcov, 78

WAIC, 76
weta, 6, 16, 46, 60, 77
wqi(qid (wqi(qid-package), 3
wqi(qid-class, 41, 43, 46, 60, 78
wqi(qid-package, 3