Package 'Bolstad'

January 20, 2025

Version 0.2.42

Date 2024-11-12

Title Functions for Elementary Bayesian Inference

Description A set of R functions and data sets for the book Introduction to Bayesian Statistics, Bolstad, W.M. (2017), John Wiley & Sons ISBN 978-1-118-09156-2.

License GPL (>= 2)

LazyData true

Depends R (>= 4.2.0)

Imports mvtnorm, methods

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author James Curran [aut, cre] (Original author)

Maintainer James Curran <j.curran@auckland.ac.nz>

Repository CRAN

Date/Publication 2024-11-12 10:20:15 UTC

Contents

`Bolstad-package` *Bolstad Functions*

Description

A set of R functions and data sets for the book Introduction to Bayesian Statistics, Bolstad, W.M. (2007), John Wiley & Sons ISBN 0-471-27020-2. Most of the package functions replicate the Minitab macros that are provided with the book. Some additional functions are provided to simplfy inference about the posterior distribution of the parameters of interest.

Details

Package:	Bolstad
Type:	Package
Version:	0.2-26
Date:	2015-05-01
License:	GPL 2

Author(s)

James Curran Maintainer: James Curran <j.curran@auckland.ac.nz> ~~ The author and/or maintainer of the package ~~

References

Bolstad, W.M. (2007), Introduction to Bayesian Statistics, John Wiley & Sons.

`as.data.frame.Bolstad` *as.data.frame.Bolstad*

Description

as.data.frame.Bolstad

Usage

```
## S3 method for class 'Bolstad'
as.data.frame(x, ...)
```

Arguments

`x`	an object of class `Bolstad`
`...`	any extra arguments needed.

`bayes.lin.reg` *Bayesian inference for simple linear regression*

Description

This function is used to find the posterior distribution of the simple linear regression slope variable β when we have a random sample of ordered pairs (x_i, y_i) from the simple linear regression model:

$$y_i = \alpha_{\bar{x}} + \beta x_i + \epsilon_i$$

where the observation errors are, ϵ_i, independent $normal(0, \sigma^2)$ with known variance.

Usage

```
bayes.lin.reg(
  y,
  x,
  slope.prior = c("flat", "normal"),
  intcpt.prior = c("flat", "normal"),
  mb0 = 0,
  sb0 = 0,
  ma0 = 0,
  sa0 = 0,
  sigma = NULL,
  alpha = 0.05,
  plot.data = FALSE,
  pred.x = NULL,
  ...
)
```

Arguments

`y`	the vector of responses.
`x`	the value of the explantory variable associated with each response.
`slope.prior`	use a "flat" prior or a "normal" prior. for β
`intcpt.prior`	use a "flat" prior or a "normal" prior. for $\alpha_[\bar{x}]$
`mb0`	the prior mean of the simple linear regression slope variable β. This argument is ignored for a flat prior.
`sb0`	the prior std. deviation of the simple linear regression slope variable β - must be greater than zero. This argument is ignored for a flat prior.
`ma0`	the prior mean of the simple linear regression intercept variable $\alpha_{\bar{x}}$. This argument is ignored for a flat prior.
`sa0`	the prior std. deviation of the simple linear regression variable $\alpha_{\bar{x}}$ - must be greater than zero. This argument is ignored for a flat prior.

sigma	the value of the std. deviation of the residuals. By default, this is assumed to be unknown and the sample value is used instead. This affects the prediction intervals.
alpha	controls the width of the credible interval.
plot.data	if true the data are plotted, and the posterior regression line superimposed on the data.
pred.x	a vector of x values for which the predicted y values are obtained and the std. errors of prediction
...	additional arguments that are passed to Bolstad.control

Value

A list will be returned with the following components:

post.coef	the posterior mean of the intecept and the slope
post.coef	the posterior standard deviation of the intercept the slope
pred.x	the vector of values for which predictions have been requested. If pred.x is NULL then this is not returned
pred.y	the vector predicted values corresponding to pred.x. If pred.x is NULL then this is not returned
pred.se	The standard errors of the predicted values in pred.y. If pred.x is NULL then this is not returned

Examples

```
## generate some data from a known model, where the true value of the
## intercept alpha is 2, the true value of the slope beta is 3, and the
## errors come from a normal(0,1) distribution
set.seed(123)
x = rnorm(50)
y = 2 + 3*x + rnorm(50)

## use the function with a flat prior for the slope beta and a
## flat prior for the intercept, alpha_xbar.

bayes.lin.reg(y,x)

## use the function with a normal(0,3) prior for the slope beta and a
## normal(30,10) prior for the intercept, alpha_xbar.

bayes.lin.reg(y,x,"n","n",0,3,30,10)

## use the same data but plot it and the credible interval

bayes.lin.reg(y,x,"n","n",0,3,30,10, plot.data = TRUE)

## The heart rate vs. O2 uptake example 14.1
O2 = c(0.47,0.75,0.83,0.98,1.18,1.29,1.40,1.60,1.75,1.90,2.23)
HR = c(94,96,94,95,104,106,108,113,115,121,131)
```

```
plot(HR,O2,xlab="Heart Rate",ylab="Oxygen uptake (Percent)")

bayes.lin.reg(O2,HR,"n","f",0,1,sigma=0.13)

## Repeat the example but obtain predictions for HR = 100 and 110

bayes.lin.reg(O2,HR,"n","f",0,1,sigma=0.13,pred.x=c(100,110))
```

`bayes.lm` *Bayesian inference for multiple linear regression*

Description

bayes.lm is used to fit linear models in the Bayesian paradigm. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance (although these are not tested). This documentation is shamelessly adapated from the lm documentation

Usage

```
bayes.lm(
  formula,
  data,
  subset,
  na.action,
  model = TRUE,
  x = FALSE,
  y = FALSE,
  center = TRUE,
  prior = NULL,
  sigma = FALSE
)
```

Arguments

`formula`	an object of class `formula` (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `bayes.lm` is called.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set by the `na.action` setting of options, and is `link[stats]{na.fail}` if that is unset. The 'factory-fresh' default is `na.omit`. Another possible value is `NULL`, no action. Value `na.exclude` can be useful.

model, x, y	logicals. If TRUE the corresponding components of the fit (the model frame, the model matrix, the response) are returned. β. This argument is ignored for a flat prior.
center	logical or numeric. If TRUE then the covariates will be centered on their means to make them orthogonal to the intercept. This probably makes no sense for models with factors, and if the argument is numeric then it contains a vector of covariate indices to be centered (not implemented yet).
prior	A list containing b0 (A vector of prior coefficients) and V0 (A prior covariance matrix)
sigma	the population standard deviation of the errors. If FALSE then this is estimated from the residual sum of squares from the ML fit.

Details

Models for bayes.lm are specified symbolically. A typical model has the form response ~ terms where response is the (numeric) response vector and terms is a series of terms which specifies a linear predictor for response. A terms specification of the form first + second indicates all the terms in first together with all the terms in second with duplicates removed. A specification of the form first:second indicates the set of terms obtained by taking the interactions of all terms in first with all terms in second. The specification first*second indicates the cross of first and second. This is the same as first + second + first:second.

See model.matrix for some further details. The terms in the formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on: to avoid this pass a terms object as the formula (see aov and demo(glm.vr) for an example).

A formula has an implied intercept term. To remove this use either y ~ x - 1 or y ~ 0 + x. See formula for more details of allowed formulae.

bayes.lm calls the lower level function lm.fit to get the maximum likelihood estimates see below, for the actual numerical computations. For programming only, you may consider doing likewise.

subset is evaluated in the same way as variables in formula, that is first in data and then in the environment of formula.

Value

bayes.lm returns an object of class Bolstad. The summary function is used to obtain and print a summary of the results much like the usual summary from a linear regression using lm. The generic accessor functions coef, fitted.values and residuals extract various useful features of the value returned by bayes.lm. Note that the residuals are computed at the posterior mean values of the coefficients.

An object of class "Bolstad" from this function is a list containing at least the following components:

coefficients	a named vector of coefficients which contains the posterior mean
post.var	a matrix containing the posterior variance-covariance matrix of the coefficients
post.sd	sigma
residuals	the residuals, that is response minus fitted values (computed at the posterior mean)
fitted.values	the fitted mean values (computed at the posterior mean)

df.residual	the residual degrees of freedom
call	the matched call
terms	the terms object used
y	if requested, the response used
x	if requested, the model matrix used
model	if requested (the default), the model frame used
na.action	(where relevant) information returned by model.frame on the special handling of NAs

Examples

```
data(bears)
bears = subset(bears, Obs.No==1)
bears = bears[,-c(1,2,3,11,12)]
bears = bears[ ,c(7, 1:6)]
bears$Sex = bears$Sex - 1
log.bears = data.frame(log.Weight = log(bears$Weight), bears[,2:7])

b0 = rep(0, 7)
V0 = diag(rep(1e6,7))

fit = bayes.lm(log(Weight)~Sex+Head.L+Head.W+Neck.G+Length+Chest.G, data = bears,
               prior = list(b0 = b0, V0 = V0))
summary(fit)
print(fit)


## Dobson (1990) Page 9: Plant Weight Data:
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)

lm.D9 <- lm(weight ~ group)
bayes.D9 <- bayes.lm(weight ~ group)

summary(lm.D9)
summary(bayes.D9)
```

---

bayes.t.test *Bayesian t-test*

Description

Performs one and two sample t-tests (in the Bayesian hypothesis testing framework) on vectors of data

Usage

```
bayes.t.test(x, ...)

## Default S3 method:
bayes.t.test(
  x,
  y = NULL,
  alternative = c("two.sided", "less", "greater"),
  mu = 0,
  paired = FALSE,
  var.equal = TRUE,
  conf.level = 0.95,
  prior = c("jeffreys", "joint.conj"),
  m = NULL,
  n0 = NULL,
  sig.med = NULL,
  kappa = 1,
  sigmaPrior = "chisq",
  nIter = 10000,
  nBurn = 1000,
  ...
)

## S3 method for class 'formula'
bayes.t.test(formula, data, subset, na.action, ...)
```

Arguments

x	a (non-empty) numeric vector of data values.
...	any additional arguments
y	an optional (non-empty) numeric vector of data values.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
mu	a number indicating the true value of the mean (or difference in means if you are performing a two sample test).
paired	a logical indicating whether you want a paired t-test.
var.equal	a logical variable indicating whether to treat the two variances as being equal. If TRUE (default) then the pooled variance is used to estimate the variance otherwise the Welch (or Satterthwaite) approximation to the degrees of freedom is used. The unequal variance case is implented using Gibbs sampling.
conf.level	confidence level of interval.
prior	a character string indicating which prior should be used for the means, must be one of "jeffreys" (default) for independent Jeffreys' priors on the unknown mean(s) and variance(s), or "joint.conj" for a joint conjugate prior.
m	if the joint conjugate prior is used then the user must specify a prior mean in the one-sample or paired case, or two prior means in the two-sample case. Note

that if the hypothesis is that there is no difference between the means in the two-sample case, then the values of the prior means should usually be equal, and if so, then their actual values are irrelvant.This parameter is not used if the user chooses a Jeffreys' prior.

`n0` if the joint conjugate prior is used then the user must specify the prior precision or precisions in the two sample case that represent our level of uncertainty about the true mean(s). This parameter is not used if the user chooses a Jeffreys' prior.

`sig.med` if the joint conjugate prior is used then the user must specify the prior median for the unknown standard deviation. This parameter is not used if the user chooses a Jeffreys' prior.

`kappa` if the joint conjugate prior is used then the user must specify the degrees of freedom for the inverse chi-squared distribution used for the unknown standard deviation. Usually the default of 1 will be sufficient. This parameter is not used if the user chooses a Jeffreys' prior.

`sigmaPrior` If a two-sample t-test with unequal variances is desired then the user must choose between using an chi-squared prior ("chisq") or a gamma prior ("gamma") for the unknown population standard deviations. This parameter is only used if `var.equal` is set to `FALSE`.

`nIter` Gibbs sampling is used when a two-sample t-test with unequal variances is desired. This parameter controls the sample size from the posterior distribution.

`nBurn` Gibbs sampling is used when a two-sample t-test with unequal variances is desired. This parameter controls the number of iterations used to burn in the chains before the procedure starts sampling in order to reduce correlation with the starting values.

`formula` a formula of the form `lhs ~ rhs` where lhs is a numeric variable giving the data values and rhs a factor with two levels giving the corresponding groups.

`data` an optional matrix or data frame (or similar: see `model.frame`) containing the variables in the formula formula. By default the variables are taken from `environment(formula)`.

`subset` currently ingored.

`na.action` currently ignored.

Value

A list with class "htest" containing the following components:

`statistic` the value of the t-statistic.

`parameter` the degrees of freedom for the t-statistic.

`p.value` the p-value for the test.

"

`conf.int` a confidence interval for the mean appropriate to the specified alternative hypothesis.

`estimate` the estimated mean or difference in means depending on whether it was a one-sample test or a two-sample test.

null.value	the specified hypothesized value of the mean or mean difference depending on whether it was a one-sample test or a two-sample test.
alternative	a character string describing the alternative hypothesis.
method	a character string indicating what type of t-test was performed.
data.name	a character string giving the name(s) of the data.
result	an object of class Bolstad

Methods (by class)

- bayes.t.test(default): Bayesian t-test
- bayes.t.test(formula): Bayesian t-test

Author(s)

R Core with Bayesian internals added by James Curran

Examples

```
bayes.t.test(1:10, y = c(7:20))      # P = .3.691e-01

## Same example but with using the joint conjugate prior
## We set the prior means equal (and it doesn't matter what the value is)
## the prior precision is 0.01, which is a prior standard deviation of 10
## we're saying the true difference of the means is between [-25.7, 25.7]
## with probability equal to 0.99. The median value for the prior on sigma is 2
## and we're using a scaled inverse chi-squared prior with 1 degree of freedom
bayes.t.test(1:10, y = c(7:20), var.equal = TRUE, prior = "joint.conj",
             m = c(0,0), n0 =  rep(0.01, 2), sig.med = 2)

##' Same example but with a large outlier. Note the assumption of equal variances isn't sensible
bayes.t.test(1:10, y = c(7:20, 200)) # P = .1979    -- NOT significant anymore

## Classical example: Student's sleep data
plot(extra ~ group, data = sleep)

## Traditional interface
with(sleep, bayes.t.test(extra[group == 1], extra[group == 2]))

## Formula interface
bayes.t.test(extra ~ group, data = sleep)
```

---

bears *bears*

Description

Body measurements for 143 wild bears.

Format

A data frame with 143 observations on the following 12 variables.

- ID. Indentification number
- Age. Bear's age, in months. Note, wild bears are always born in January, so an expert can estimate the bear's age without directly asking it how old it is.
- Month. Month when the measurement was made. 1 = Jan., 12 = Dec. Since bears hibernate in the winter, their body shape probably depends on the season.
- Sex. 1 = male 2 = female
- Head.L. Length of the head, in inches
- Head.W. Width of the head, in inches
- Neck.G. Girth (distance around) the neck, in inches
- Length. Body length, in inches
- Chest.G. Girth (distance around) the chest, in inches
- Weight. Weight of the bear, in pounds
- Obs.No. Observation number for this bear. For example, the bear with ID = 41 (Bertha) was measured on four occasions, in the months coded 7, 8, 11, and 5. The value of Obs.No goes from 1 to 4 for these observations.
- Name. The names of the bears given to them by the researchers

Details

Wild bears were anesthetized, and their bodies were measured and weighed. One goal of the study was to make a table (or perhaps a set of tables) for hunters, so they could estimate the weight of a bear based on other measurements. This would be used because in the forest it is easier to measure the length of a bear, for example, than it is to weigh it.

Source

This data is in the example data set Bears.MTW distributed with Minitab

References

This data set was supplied by Gary Alt. Entertaining references are in Reader's Digest April, 1979, and Sports Afield September, 1981.

Examples

```
data(bears)
boxplot(Weight~Sex, data = bears)
```

binobp *Binomial sampling with a beta prior*

Description

Evaluates and plots the posterior density for π, the probability of a success in a Bernoulli trial, with binomial sampling and a continous $beta(a, b)$ prior.

Usage

```
binobp(x, n, a = 1, b = 1, pi = seq(0, 1, by = 0.001), ...)
```

Arguments

x	the number of observed successes in the binomial experiment.
n	the number of trials in the binomial experiment.
a	parameter for the beta prior - must be greater than zero
b	parameter for the beta prior - must be greater than zero
pi	A range of values for the prior to be calculated over.
...	additional arguments that are passed to `Bolstad.control`

Value

An object of class 'Bolstad' is returned. This is a list with the following components:

prior	the prior density of π, i.e. the $beta(a, b)$ density
likelihood	the likelihood of x given π and n, i.e. the $binomial(n, \pi)$ density
posterior	the posterior density of π given x and n - i.e. the $beta(a + x, b + n - x)$ density
pi	the values of π for which the posterior density was evaluated
mean	the posterior mean
var	the posterior variance
sd	the posterior std. deviation
quantiles	a set of quantiles from the posterior
cdf	a cumulative distribution function for the posterior
quantileFun	a quantile function for the posterior

See Also

binodp binogcp

Examples

```
## simplest call with 6 successes observed in 8 trials and a beta(1,1) uniform
## prior
binobp(6,8)

## 6 successes observed in 8 trials and a non-uniform beta(0.5,6) prior
binobp(6,8,0.5,6)

## 4 successes observed in 12 trials with a non uniform beta(3,3) prior
## plot the stored prior, likelihood and posterior
results = binobp(4, 12, 3, 3)
decomp(results)
```

---

binodp *Binomial sampling with a discrete prior*

Description

Evaluates and plots the posterior density for π, the probability of a success in a Bernoulli trial, with binomial sampling and a discrete prior on π

Usage

```
binodp(x, n, pi = NULL, pi.prior = NULL, n.pi = 10, ...)
```

Arguments

x	the number of observed successes in the binomial experiment.
n	the number of trials in the binomial experiment.
pi	a vector of possibilities for the probability of success in a single trial. if pi is NULL then a discrete uniform prior for π will be used.
pi.prior	the associated prior probability mass.
n.pi	the number of possible π values in the prior
...	additional arguments that are passed to Bolstad.control

Value

A list will be returned with the following components:

pi	the vector of possible π values used in the prior
pi.prior	the associated probability mass for the values in π
likelihood	the scaled likelihood function for π given x and n
posterior	the posterior probability of π given x and n
f.cond	the conditional distribution of x given π and n
f.joint	the joint distribution of x and π given n
f.marg	the marginal distribution of x

See Also

binobp binogcp

Examples

```
## simplest call with 6 successes observed in 8 trials and a uniform prior
binodp(6,8)

## same as previous example but with more possibilities for pi
binodp(6, 8, n.pi = 100)

## 6 successes, 8 trials and a non-uniform discrete prior
pi = seq(0, 1, by = 0.01)
pi.prior = runif(101)
pi.prior = sort(pi.prior / sum(pi.prior))
binodp(6, 8, pi, pi.prior)

## 5 successes, 6 trials, non-uniform prior
pi = c(0.3, 0.4, 0.5)
pi.prior = c(0.2, 0.3, 0.5)
results = binodp(5, 6, pi, pi.prior)

## plot the results from the previous example using a side-by-side barplot
results.matrix = rbind(results$pi.prior,results$posterior)
colnames(results.matrix) = pi
barplot(results.matrix, col = c("red", "blue"), beside = TRUE,
      xlab = expression(pi), ylab=expression(Probability(pi)))
box()
legend("topleft", bty = "n", cex = 0.7,
       legend = c("Prior", "Posterior"), fill = c("red", "blue"))
```

---

binogcp *Binomial sampling with a general continuous prior*

Description

Evaluates and plots the posterior density for π, the probability of a success in a Bernoulli trial, with binomial sampling and a general continuous prior on π

Usage

```
binogcp(
  x,
  n,
  density = c("uniform", "beta", "exp", "normal", "user"),
  params = c(0, 1),
  n.pi = 1000,
  pi = NULL,
```

```
  pi.prior = NULL,
  ...
)
```

Arguments

x	the number of observed successes in the binomial experiment.
n	the number of trials in the binomial experiment.
density	may be one of "beta", "exp", "normal", "student", "uniform" or "user"
params	if density is one of the parameteric forms then then a vector of parameters must be supplied. beta: a, b exp: rate normal: mean, sd uniform: min, max
n.pi	the number of possible π values in the prior
pi	a vector of possibilities for the probability of success in a single trial. This must be set if density = "user".
pi.prior	the associated prior probability mass. This must be set if density = "user".
...	additional arguments that are passed to Bolstad.control

Value

A list will be returned with the following components:

likelihood	the scaled likelihood function for π given x and n
posterior	the posterior probability of π given x and n
pi	the vector of possible π values used in the prior
pi.prior	the associated probability mass for the values in π

See Also

binobp binodp

Examples

```
## simplest call with 6 successes observed in 8 trials and a continuous
## uniform prior
binogcp(6, 8)

## 6 successes, 8 trials and a Beta(2, 2) prior
binogcp(6, 8,density = "beta", params = c(2, 2))

## 5 successes, 10 trials and a N(0.5, 0.25) prior
binogcp(5, 10, density = "normal", params = c(0.5, 0.25))

## 4 successes, 12 trials with a user specified triangular continuous prior
pi = seq(0, 1,by = 0.001)
pi.prior = rep(0, length(pi))
priorFun = createPrior(x = c(0, 0.5, 1), wt = c(0, 2, 0))
pi.prior = priorFun(pi)
results = binogcp(4, 12, "user", pi = pi, pi.prior = pi.prior)
```

```
## find the posterior CDF using the previous example and Simpson's rule
myCdf = cdf(results)
plot(myCdf, type = "l", xlab = expression(pi[0]),
   ylab = expression(Pr(pi <= pi[0])))

## use the quantile function to find the 95% credible region.
qtls = quantile(results, probs = c(0.025, 0.975))
cat(paste("Approximate 95% credible interval : ["
, round(qtls[1], 4), " ", round(qtls, 4), "]\n", sep = ""))

## find the posterior mean, variance and std. deviation
## using the output from the previous example
post.mean = mean(results)
post.var = var(results)
post.sd = sd(results)

# calculate an approximate 95% credible region using the posterior mean and
# std. deviation
lb = post.mean - qnorm(0.975) * post.sd
ub = post.mean + qnorm(0.975) * post.sd

cat(paste("Approximate 95% credible interval : ["
, round(lb, 4), " ", round(ub, 4), "]\n", sep = ""))
```

`binomixp` *Binomial sampling with a beta mixture prior*

Description

Evaluates and plots the posterior density for π, the probability of a success in a Bernoulli trial, with binomial sampling when the prior density for π is a mixture of two beta distributions, $beta(a_0, b_0)$ and $beta(a_1, b_1)$.

Usage

```
binomixp(x, n, alpha0 = c(1, 1), alpha1 = c(1, 1), p = 0.5, ...)
```

Arguments

`x`	the number of observed successes in the binomial experiment.
`n`	the number of trials in the binomial experiment.
`alpha0`	a vector of length two containing the parameters, a_0 and b_0, for the first component beta prior - must be greater than zero. By default the elements of alpha0 are set to 1.
`alpha1`	a vector of length two containing the parameters, a_1 and b_1, for the second component beta prior - must be greater than zero. By default the elements of alpha1 are set to 1.

p	The prior mixing proportion for the two component beta priors. That is the prior is $pimesbeta(a_0, b_0) + (1 - p)imesbeta(a_1, b_1)$. p is set to 0.5 by default
...	additional arguments that are passed to `Bolstad.control`

Value

A list will be returned with the following components:

pi	the values of π for which the posterior density was evaluated
posterior	the posterior density of π given n and x
likelihood	the likelihood function for π given x and n, i.e. the $binomial(n, \pi)$ density
prior	the prior density of π density

See Also

binodp binogcp normmixp

Examples

```
## simplest call with 6 successes observed in 8 trials and a 50:50 mix
## of two beta(1,1) uniform priors
binomixp(6,8)

## 6 successes observed in 8 trials and a 20:80 mix of a non-uniform
## beta(0.5,6) prior and a uniform beta(1,1) prior
binomixp(6,8,alpha0=c(0.5,6),alpha1=c(1,1),p=0.2)

## 4 successes observed in 12 trials with a 90:10 non uniform beta(3,3) prior
## and a non uniform beta(4,12).
## Plot the stored prior, likelihood and posterior
results = binomixp(4, 12, c(3, 3), c(4, 12), 0.9)$mix

par(mfrow = c(3,1))
y.lims = c(0, 1.1 * max(results$posterior, results$prior))

plot(results$pi,results$prior,ylim=y.lims,type='l'
,xlab=expression(pi),ylab='Density',main='Prior')
polygon(results$pi,results$prior,col='red')

plot(results$pi,results$likelihood,type='l',
     xlab = expression(pi), ylab = 'Density', main = 'Likelihood')
polygon(results$pi,results$likelihood,col='green')

plot(results$pi,results$posterior,ylim=y.lims,type='l'
,xlab=expression(pi),ylab='Density',main='Posterior')
polygon(results$pi,results$posterior,col='blue')
```

Bolstad.control *Control Bolstad functions*

Description

Control Bolstad functions

Usage

```
Bolstad.control(plot = TRUE, quiet = FALSE, ...)
```

Arguments

plot	if TRUE then draw a plot (for functions that actually have plots)
quiet	if TRUE then suppress the function output
...	additional parameters

Value

an invisible list of options and their values

cdf *Cumulative distribution function generic*

Description

This function returns the cumulative distribution function (cdf) of the posterior distribution of the parameter interest over the range of values for which the posterior is specified.

Usage

```
cdf(x, ...)

## S3 method for class 'Bolstad'
cdf(x, ...)
```

Arguments

x	An object for which we want to compute the cdf
...	Any other parameters. Not currently used.

Value

either the exact cdf of the posterior if a conjugate prior has been used, or a a stats::splinefun which will compute the lower tail probability of the parameter for any valid input.

Methods (by class)

- `cdf(Bolstad)`: Cumulative distribution function for posterior density

Author(s)

James Curran

`createPrior` *Create prior generic*

Description

Create prior generic

Usage

```
createPrior(x, ...)
```

Arguments

x	a vector of x values at which the prior is to be specified (the support of the prior).
...	optional exta arguments. Not currently used.

Value

a linear interpolation function where the weights have been scaled so the function (numerically) integrates to 1.

`createPrior.default` *Create prior default method*

Description

Create prior default method

Usage

```
## Default S3 method:
createPrior(x, wt, ...)
```

Arguments

x	a vector of x values at which the prior is to be specified (the support of the prior). This should contain unique values in ascending order. The function will sort values if x is unsorted with a warning, and will halt if x contains any duplicates or negative lag 1 differences.
wt	a vector of weights corresponding to the weight of the prior at the given x values.
...	optional exta arguments. Not currently used.

Value

a linear interpolation function where the weights have been scaled so the function (numerically) integrates to 1.

---

decomp *Plot the prior, likelihood, and posterior on the same plot.*

Description

This function takes any object of class `Bolstad` and plots the prior, likelihood and posterior on the same plot. The aim is to show the influence of the prior, and the likelihood on the posterior.

Usage

```
decomp(x, ...)
```

Arguments

x	an object of class `Bolstad`.
...	any other arguments to be passed to the `plot` function.

Note

Note that `xlab`, `ylab`, `main`, `axes`, `xlim`, `ylim` and `type` are all used in the function so specifying them is unlikely to have any effect.

Author(s)

James Curran

Examples

```
# an example with a binomial sampling situation
results = binobp(4, 12, 3, 3, plot = FALSE)
decomp(results)

# an example with normal data
y = c(2.99,5.56,2.83,3.47)
results = normnp(y, 3, 2, 1, plot = FALSE)
decomp(results)
```

IQR *Interquartile Range generic*

Description

Compute the interquartile range.

Usage

```
IQR(x, ...)
```

Arguments

x	an object.
...	any additional arguments. These are primarily used in `IQR.default` which calls `stats::IQR`.

Details

If `x` is an object of class `Bolstad` then the posterior IQR of the parameter of interest will be calculated.

Author(s)

James Curran

lines.Bolstad *Lines method for Bolstad objects*

Description

Allows simple addition of posterior distributions from other results to an existing plot

Usage

```
## S3 method for class 'Bolstad'
lines(x, ...)
```

Arguments

x	an object of class Bolstad.
...	any additional parameters to be passed to graphics::lines.

mean.Bolstad *Calculate the posterior mean*

Description

Calculate the posterior mean of an object of class Bolstad. If the object has a member mean then it will return this value otherwise it will calculate $\int_{-\infty}^{+\infty} \theta f(\theta|x).d\theta$ using linear interpolation to approximate the density function and numerical integration where θ is the variable for which we want to do Bayesian inference, and x is the data.

Usage

```
## S3 method for class 'Bolstad'
mean(x, ...)
```

Arguments

x	An object of class Bolstad
...	Any other arguments. This parameter is currently ignored but it could be useful in the future to deal with problematic data.

Value

The posterior mean of the variable of inference given the data.

Examples

```
# The useful of this method is really highlighted when we have a general
# continuous prior. In this example we are interested in the posterior mean of
# an normal mean. Our prior is triangular over [-3, 3]
set.seed(123)
x = rnorm(20, -0.5, 1)
mu = seq(-3, 3, by = 0.001)
mu.prior = rep(0, length(mu))
mu.prior[mu <= 0] = 1 / 3 + mu[mu <= 0] / 9
mu.prior[mu > 0] = 1 / 3 - mu[mu > 0] / 9
results = normgcp(x, 1, density = "user", mu = mu, mu.prior = mu.prior)
mean(results)
```

---

median.Bolstad *Median generic*

Description

Compute the posterior median of the posterior distribution

Usage

```
## S3 method for class 'Bolstad'
median(x, na.rm = FALSE, ...)
```

Arguments

x	an object.
na.rm	Ideally if TRUE then missing values will be removed, but not currently used.
...	[>=R3.4.0 only] Not currently used.

Details

If x is an object of class Bolstad then the posterior median of the parameter of interest will be calculated.

Author(s)

James Curran

moisture.df *Moisture data*

Description

Moisture level at two stages in a food manufacturing process, in-process and final. These data are given in Example 14.1

Usage

```
moisture.df
```

Format

A data frame with 25 observations on the following 6 variables.

- batch. the batch number of the measurement
- proc.level. the in-process moisture level
- final.level. natural the final moisture level of the batch
- ls.fit the least squares fitted value of final.level given proc.level
- residiual. the least squares residual
- residiual.sq. the squaredvleast squares residual

Examples

```
data(moisture.df)
plot(final.level~proc.level, data = moisture.df)
```

mvnmvnp *Bayesian inference on a mutlivariate normal (MVN) mean with a multivariate normal (MVN) prior*

Description

Evaluates posterior density for μ, the mean of a MVN distribution, with a MVN prior on μ

Usage

```
mvnmvnp(y, m0 = 0, V0 = 1, Sigma = NULL, ...)
```

Arguments

y	a vector of observations from a MVN distribution with unknown mean and known variance-covariance.
m0	the mean vector of the MVN prior, or a scalar constant so that the prior vector of length k with the same element repeated k times, e.g. `m0 = 0`
V0	the variance-covariance matrix of the MVN prior, or the diagonal of the variance-covariance matrix of the MVN prior, or a scalar constant, say n_0, so the prior is $n_0 \times \mathbf{I}_k$ where $\mathbf{I}_k$ is the k by k identity matrix.
Sigma	the known variance covariance matrix of the data. If this value is NULL, which it is by default, then the sample covariance is used. NOTE: if this is the case then the cdf and quantile functions should really be multivariate t, but they are not - in which case the results are only (approximately) valid for large samples.
...	any other values to be passed to Bolstad.control

Value

A list will be returned with the following components:

mean	the posterior mean of the MVN posterior distribution
var	the posterior variance-covariance matrix of the MVN posterior distribution
cdf	a function that will evaluation the posterior cdf at a given point. This function calls `mvtnmorm::pmvnorm`.
quantile	a function that will find quantiles from the posterior given input probabilities. This function calls `mvtnorm::qmvnorm`.

---

normdp *Bayesian inference on a normal mean with a discrete prior*

Description

Evaluates and plots the posterior density for μ, the mean of a normal distribution, with a discrete prior on μ

Usage

```
normdp(x, sigma.x = NULL, mu = NULL, mu.prior = NULL, n.mu = 50, ...)
```

Arguments

x	a vector of observations from a normal distribution with unknown mean and known std. deviation.
sigma.x	the population std. deviation of the normal distribution
mu	a vector of possibilities for the probability of success in a single trial. If mu is NULL then a uniform prior is used.
mu.prior	the associated prior probability mass.
n.mu	the number of possible μ values in the prior
...	additional arguments that are passed to `Bolstad.control`

Value

A list will be returned with the following components:

mu	the vector of possible μ values used in the prior
mu.prior	the associated probability mass for the values in μ
likelihood	the scaled likelihood function for μ given x and σ_x
posterior	the posterior probability of μ given x and σ_x

See Also

normnp normgcp

Examples

```
## generate a sample of 20 observations from a N(-0.5,1) population
x = rnorm(20,-0.5,1)

## find the posterior density with a uniform prior on mu
normdp(x,1)

## find the posterior density with a non-uniform prior on mu
mu = seq(-3,3,by=0.1)
mu.prior = runif(length(mu))
mu.prior = sort(mu.prior/sum(mu.prior))
normdp(x,1,mu,mu.prior)

## Let mu have the discrete distribution with 5 possible
## values, 2, 2.5, 3, 3.5 and 4, and associated prior probability of
## 0.1, 0.2, 0.4, 0.2, 0.1 respectively. Find the posterior
## distribution after a drawing random sample of n = 5 observations
## from a N(mu,1) distribution y = [1.52, 0.02, 3.35, 3.49, 1.82]
mu = seq(2,4,by=0.5)
mu.prior = c(0.1,0.2,0.4,0.2,0.1)
y = c(1.52,0.02,3.35,3.49,1.82)
normdp(y,1,mu,mu.prior)
```

---

normgcp *Bayesian inference on a normal mean with a general continuous prior*

Description

Evaluates and plots the posterior density for μ, the mean of a normal distribution, with a general continuous prior on μ

Usage

```
normgcp(
  x,
  sigma.x = NULL,
  density = c("uniform", "normal", "flat", "user"),
  params = NULL,
  n.mu = 50,
  mu = NULL,
  mu.prior = NULL,
  ...
)
```

Arguments

x	a vector of observations from a normal distribution with unknown mean and known std. deviation.
sigma.x	the population std. deviation of the normal distribution
density	distributional form of the prior density can be one of: "normal", "unform", or "user".
params	if density = "normal" then params must contain at least a mean and possible a std. deviation. If a std. deviation is not specified then sigma.x will be used as the std. deviation of the prior. If density = "uniform" then params must contain a minimum and a maximum value for the uniform prior. If a maximum and minimum are not specified then a $U[0,1]$ prior is used
n.mu	the number of possible μ values in the prior
mu	a vector of possibilities for the probability of success in a single trial. Must be set if density="user"
mu.prior	the associated prior density. Must be set if density="user"
...	additional arguments that are passed to Bolstad.control

Value

A list will be returned with the following components:

likelihood	the scaled likelihood function for μ given x and σ_x
posterior	the posterior probability of μ given x and σ
mu	the vector of possible μ values used in the prior
mu.prior	the associated probability mass for the values in μ

See Also

normdp normnp

Examples

```
## generate a sample of 20 observations from a N(-0.5,1) population
x = rnorm(20,-0.5,1)

## find the posterior density with a uniform U[-3,3] prior on mu
normgcp(x, 1, params = c(-3, 3))

## find the posterior density with a non-uniform prior on mu
mu = seq(-3, 3, by = 0.1)
mu.prior = rep(0, length(mu))
mu.prior[mu <= 0] = 1 / 3 + mu[mu <= 0] /9
mu.prior[mu > 0] = 1 / 3 - mu[mu > 0] / 9
normgcp(x, 1, density = "user", mu = mu, mu.prior = mu.prior)

## find the CDF for the previous example and plot it
## Note the syntax for sintegral has changed
results = normgcp(x,1,density="user",mu=mu,mu.prior=mu.prior)
cdf = sintegral(mu,results$posterior,n.pts=length(mu))$cdf
plot(cdf,type="l",xlab=expression(mu[0])
             ,ylab=expression(Pr(mu<=mu[0])))

## use the CDF for the previous example to find a 95%
## credible interval for mu. Thanks to John Wilkinson for this simplified code

lcb = cdf$x[with(cdf,which.max(x[y<=0.025]))]
ucb = cdf$x[with(cdf,which.max(x[y<=0.975]))]
cat(paste("Approximate 95% credible interval : ["
           ,round(lcb,4)," ",round(ucb,4),"]\n",sep=""))

## use the CDF from the previous example to find the posterior mean
## and std. deviation
dens = mu*results$posterior
post.mean = sintegral(mu,dens)$value

dens = (mu-post.mean)^2*results$posterior
post.var = sintegral(mu,dens)$value
post.sd = sqrt(post.var)

## use the mean and std. deviation from the previous example to find
## an approximate 95% credible interval
lb = post.mean-qnorm(0.975)*post.sd
ub = post.mean+qnorm(0.975)*post.sd


cat(paste("Approximate 95% credible interval : ["
   ,round(lb,4)," ",round(ub,4),"]\n",sep=""))

## repeat the last example but use the new summary functions for the posterior
results = normgcp(x, 1, density = "user", mu = mu, mu.prior = mu.prior)

## use the cdf function to get the cdf and plot it
postCDF = cdf(results) ## note this is a function
```

```
plot(results$mu, postCDF(results$mu), type="l", xlab = expression(mu[0]),
     ylab = expression(Pr(mu <= mu[0])))

## use the quantile function to get a 95% credible interval
ci = quantile(results, c(0.025, 0.975))
ci

## use the mean and sd functions to get the posterior mean and standard deviation
postMean = mean(results)
postSD = sd(results)
postMean
postSD

## use the mean and std. deviation from the previous example to find
## an approximate 95% credible interval
ciApprox = postMean + c(-1,1) * qnorm(0.975) * postSD
ciApprox
```

---

normmixp *Bayesian inference on a normal mean with a mixture of normal priors*

Description

Evaluates and plots the posterior density for μ, the mean of a normal distribution, with a mixture of normal priors on μ

Usage

```
normmixp(
  x,
  sigma.x,
  prior0,
  prior1,
  p = 0.5,
  mu = NULL,
  n.mu = max(100, length(mu)),
  ...
)
```

Arguments

x	a vector of observations from a normal distribution with unknown mean and known std. deviation.
sigma.x	the population std. deviation of the observations.
prior0	the vector of length 2 which contains the means and standard deviation of your precise prior.

prior1	the vector of length 2 which contains the means and standard deviation of your vague prior.
p	the mixing proportion for the two component normal priors.
mu	a vector of prior possibilities for the mean. If it is NULL, then a vector centered on the sample mean is created.
n.mu	the number of possible μ values in the prior.
...	additional arguments that are passed to Bolstad.control

Value

A list will be returned with the following components:

mu	the vector of possible μ values used in the prior
prior	the associated probability mass for the values in μ
likelihood	the scaled likelihood function for μ given x and σ_x
posterior	the posterior probability of μ given x and σ_x

See Also

binomixp normdp normgcp

Examples

```
## generate a sample of 20 observations from a N(-0.5, 1) population
x = rnorm(20, -0.5, 1)

## find the posterior density with a N(0, 1) prior on mu - a 50:50 mix of
## two N(0, 1) densities
normmixp(x, 1, c(0, 1), c(0, 1))

## find the posterior density with 50:50 mix of a N(0.5, 3) prior and a
## N(0, 1) prior on mu
normmixp(x, 1, c(0.5, 3), c(0, 1))

## Find the posterior density for mu, given a random sample of 4
## observations from N(mu, 1), y = [2.99, 5.56, 2.83, 3.47],
## and a 80:20 mix of a N(3, 2) prior and a N(0, 100) prior for mu
x = c(2.99, 5.56, 2.83, 3.47)
normmixp(x, 1, c(3, 2), c(0, 100), 0.8)
```

normnp *Bayesian inference on a normal mean with a normal prior*

Description

Evaluates and plots the posterior density for μ, the mean of a normal distribution, with a normal prior on μ

Usage

```
normnp(
  x,
  m.x = 0,
  s.x = 1,
  sigma.x = NULL,
  mu = NULL,
  n.mu = max(100, length(mu)),
  ...
)
```

Arguments

x	a vector of observations from a normal distribution with unknown mean and known std. deviation.
m.x	the mean of the normal prior
s.x	the standard deviation of the normal prior
sigma.x	the population std. deviation of the normal distribution. If this value is NULL, which it is by default, then a flat prior is used and m.x and s.x are ignored
mu	a vector of prior possibilities for the true mean. If this is null, then a set of values centered on the sample mean is used.
n.mu	the number of possible μ values in the prior
...	optional control arguments. See Bolstad.control

Value

A list will be returned with the following components:

mu	the vector of possible μ values used in the prior
mu.prior	the associated probability mass for the values in μ
likelihood	the scaled likelihood function for μ given x and σ_x
posterior	the posterior probability of μ given x and σ_x
mean	the posterior mean
sd	the posterior standard deviation
qtls	a selection of quantiles from the posterior density

See Also

normdp normgcp

Examples

```
## generate a sample of 20 observations from a N(-0.5,1) population
x = rnorm(20,-0.5,1)

## find the posterior density with a N(0,1) prior on mu
normnp(x,sigma=1)

## find the posterior density with N(0.5,3) prior on mu
normnp(x,0.5,3,1)

## Find the posterior density for mu, given a random sample of 4
## observations from N(mu,sigma^2=1), y = [2.99, 5.56, 2.83, 3.47],
## and a N(3,sd=2)$ prior for mu
y = c(2.99,5.56,2.83,3.47)
normnp(y,3,2,1)
```

---

nvaricp *Bayesian inference for a normal standard deviation with a scaled inverse chi-squared distribution*

Description

Evaluates and plots the posterior density for σ, the standard deviation of a Normal distribution where the mean μ is known

Usage

```
nvaricp(y, mu, S0, kappa, ...)
```

Arguments

y	a random sample from a $normal(\mu, \sigma^2)$ distribution.
mu	the known population mean of the random sample.
S0	the prior scaling factor.
kappa	the degrees of freedom of the prior.
...	additional arguments that are passed to Bolstad.control

Value

A list will be returned with the following components:

sigma	the vaules of σ for which the prior, likelihood and posterior have been calculated
prior	the prior density for σ
likelihood	the likelihood function for σ given y
posterior	the posterior density of μ given y
S1	the posterior scaling constant
kappa1	the posterior degrees of freedom

Examples

```
## Suppose we have five observations from a normal(mu, sigma^2)
## distribution mu = 200 which are 206.4, 197.4, 212.7, 208.5.
y = c(206.4, 197.4, 212.7, 208.5, 203.4)

## We wish to choose a prior that has a median of 8. This happens when
## S0 = 29.11 and kappa = 1
nvaricp(y,200,29.11,1)

##  Same as the previous example but a calculate a 95% credible
## interval for sigma. NOTE this method has changed
results = nvaricp(y,200,29.11,1)
quantile(results, probs = c(0.025, 0.975))
```

---

plot.Bolstad *Plot method for objects of type Bolstad*

Description

A unified plotting method for plotting the prior, likelihood and posterior from any of the analyses in the book

Usage

```
## S3 method for class 'Bolstad'
plot(
  x,
  overlay = TRUE,
  which = c(1, 3),
  densCols = c("red", "green", "blue")[which],
  legendLoc = "topleft",
  scaleLike = FALSE,
  xlab = eval(expression(x$name)),
  ylab = "",
  main = "Shape of prior and posterior",
```

```
  ylim = c(0, max(cbind(x$prior, x$likelihood, x$posterior)[, which]) * 1.1),
  cex = 0.7,
  ...
)
```

Arguments

x	A S3 object of class Bolstad
overlay	if FALSE then up to three plots will be drawn side-by-side
which	Control which of the prior = 1, likelihood = 2, and posterior = 3, are plots. This is set to prior and posterior by default to retain compatibility with the book
densCols	The colors of the lines for each of the prior, likelihood and posterior
legendLoc	The location of the legend, usually either "topright" or "topleft"
scaleLike	If TRUE, then the likelihood will be scaled to have approximately the same maximum value as the posterior
xlab	Label for x axis
ylab	Label for y axis
main	Title of plot
ylim	Vector giving y coordinate range
cex	Character expansion multiplier
...	Any remaining arguments are fed to the plot command

Details

The function provides a unified way of plotting the prior, likelihood and posterior from any of the functions in the library that return these quantities. It will produce an overlay of the lines by default, or separate panels if overlay = FALSE.

Author(s)

James Curran

Examples

```
x = rnorm(20,-0.5,1)
## find the posterior density with a N(0,1) prior on mu
b = normnp(x,sigma=1)
plot(b)
plot(b, which = 1:3)
plot(b, overlay = FALSE, which = 1:3)
```

poisdp *Poisson sampling with a discrete prior*

Description

Evaluates and plots the posterior density for μ, the mean rate of occurance in a Poisson process and a discrete prior on μ

Usage

```
poisdp(y.obs, mu, mu.prior, ...)
```

Arguments

y.obs	a random sample from a Poisson distribution.
mu	a vector of possibilities for the mean rate of occurance of an event over a finite period of space or time.
mu.prior	the associated prior probability mass.
...	additional arguments that are passed to Bolstad.control

Value

A list will be returned with the following components:

likelihood	the scaled likelihood function for μ given y_{obs}
posterior	the posterior probability of μ given y_{obs}
mu	the vector of possible μ values used in the prior
mu.prior	the associated probability mass for the values in μ

See Also

poisgamp poisgcp

Examples

```
## simplest call with an observation of 4 and a uniform prior on the
## values mu = 1,2,3
poisdp(4,1:3,c(1,1,1)/3)

##  Same as the previous example but a non-uniform discrete prior
mu = 1:3
mu.prior = c(0.3,0.4,0.3)
poisdp(4,mu=mu,mu.prior=mu.prior)

##  Same as the previous example but a non-uniform discrete prior
mu = seq(0.5,9.5,by=0.05)
mu.prior = runif(length(mu))
```

```
mu.prior = sort(mu.prior/sum(mu.prior))
poisdp(4,mu=mu,mu.prior=mu.prior)

## A random sample of 50 observations from a Poisson distribution with
## parameter mu = 3 and  non-uniform prior
y.obs = rpois(50,3)
mu = c(1:5)
mu.prior = c(0.1,0.1,0.05,0.25,0.5)
results = poisdp(y.obs, mu, mu.prior)

##  Same as the previous example but a non-uniform discrete prior
mu = seq(0.5,5.5,by=0.05)
mu.prior = runif(length(mu))
mu.prior = sort(mu.prior/sum(mu.prior))
y.obs = rpois(50,3)
poisdp(y.obs,mu=mu,mu.prior=mu.prior)
```

`poisgamp` *Poisson sampling with a gamma prior*

Description

Evaluates and plots the posterior density for μ, the mean rate of occurance in a Poisson process and a $gamma$ prior on μ

Usage

```
poisgamp(y, shape, rate = 1, scale = 1/rate, alpha = 0.05, ...)
```

Arguments

`y`	a random sample from a Poisson distribution.
`shape`	the shape parameter of the $gamma$ prior.
`rate`	the rate parameter of the $gamma$ prior. Note that the scale is $1/rate$
`scale`	the scale parameter of the $gamma$ prior
`alpha`	the width of the credible interval is controlled by the parameter alpha.
`...`	additional arguments that are passed to `Bolstad.control`

Value

An object of class 'Bolstad' is returned. This is a list with the following components:

`prior`	the prior density assigned to μ
`likelihood`	the scaled likelihood function for μ given y
`posterior`	the posterior probability of μ given y
`shape`	the shape parameter for the $gamma$ posterior
`rate`	the rate parameter for the $gamma$ posterior

See Also

poisdp poisgcp

Examples

```
## simplest call with an observation of 4 and a gamma(1, 1), i.e. an exponential prior on the
## mu
poisgamp(4, 1, 1)

##  Same as the previous example but a gamma(10, ) prior
poisgamp(4, 10, 1)

##  Same as the previous example but an improper gamma(1, ) prior
poisgamp(4, 1, 0)

## A random sample of 50 observations from a Poisson distribution with
## parameter mu = 3 and  gamma(6,3) prior
set.seed(123)
y = rpois(50,3)
poisgamp(y,6,3)

## In this example we have a random sample from a Poisson distribution
## with an unknown mean. We will use a gamma(6,3) prior to obtain the
## posterior gamma distribution, and use the R function qgamma to get a
## 95% credible interval for mu
y = c(3,4,4,3,3,4,2,3,1,7)
results = poisgamp(y,6,3)
ci = qgamma(c(0.025,0.975),results$shape, results$rate)
cat(paste("95% credible interval for mu: [",round(ci[1],3), ",", round(ci[2],3)),"]\n")

## In this example we have a random sample from a Poisson distribution
## with an unknown mean. We will use a gamma(6,3) prior to obtain the
## posterior gamma distribution, and use the R function qgamma to get a
## 95% credible interval for mu
y = c(3,4,4,3,3,4,2,3,1,7)
results = poisgamp(y, 6, 3)
ci = quantile(results, c(0.025, 0.975))
cat(paste("95% credible interval for mu: [",round(ci[1],3), ",", round(ci[2],3)),"]\n")
```

---

poisgcp *Poisson sampling with a general continuous prior*

Description

Evaluates and plots the posterior density for μ, the mean rate of occurance of an event or objects, with Poisson sampling and a general continuous prior on μ

Usage

```
poisgcp(
  y,
  density = c("normal", "gamma", "user"),
  params = c(0, 1),
  n.mu = 100,
  mu = NULL,
  mu.prior = NULL,
  print.sum.stat = FALSE,
  alpha = 0.05,
  ...
)
```

Arguments

y	A random sample of one or more observations from a Poisson distribution
density	may be one of "gamma", "normal", or "user"
params	if density is one of the parameteric forms then then a vector of parameters must be supplied. gamma: a0,b0 normal: mean,sd
n.mu	the number of possible μ values in the prior. This number must be greater than or equal to 100. It is ignored when density="user".
mu	either a vector of possibilities for the mean of a Poisson distribution, or a range (a vector of length 2) of values. This must be set if density = "user". If mu is a range, then n.mu will be used to decide how many points to discretise this range over.
mu.prior	either a vector containing y values correspoding to the values in mu, or a function. This is used to specifiy the prior $f(\mu)$. So mu.prior can be a vector containing $f(\mu_i)$ for every μ_i, or a funtion. This must be set if density == "user".
print.sum.stat	if set to TRUE then the posterior mean, posterior variance, and a credible interval for the mean are printed. The width of the credible interval is controlled by the parameter alpha.
alpha	The width of the credible interval is controlled by the parameter alpha.
...	additional arguments that are passed to Bolstad.control

Value

A list will be returned with the following components:

mu	the vector of possible μ values used in the prior
mu.prior	the associated probability mass for the values in μ
likelihood	the scaled likelihood function for μ given y
posterior	the posterior probability of μ given y

See Also

poisdp poisgamp

Examples

```
## Our data is random sample is 3, 4, 3, 0, 1. We will try a normal
## prior with a mean of 2 and a standard deviation of 0.5.
y = c(3,4,3,0,1)
poisgcp(y, density = "normal", params = c(2,0.5))

## The same data as above, but with a gamma(6,8) prior
y = c(3,4,3,0,1)
poisgcp(y, density = "gamma", params = c(6,8))

## The same data as above, but a user specified continuous prior.
## We will use print.sum.stat to get a 99% credible interval for mu.
y = c(3,4,3,0,1)
mu = seq(0,8,by=0.001)
mu.prior = c(seq(0,2,by=0.001),rep(2,1999),seq(2,0,by=-0.0005))/10
poisgcp(y,"user",mu=mu,mu.prior=mu.prior,print.sum.stat=TRUE,alpha=0.01)

## find the posterior CDF using the results from the previous example
## and Simpson's rule. Note that the syntax of sintegral has changed.
results = poisgcp(y,"user",mu=mu,mu.prior=mu.prior)
cdf = sintegral(mu,results$posterior,n.pts=length(mu))$cdf
plot(cdf,type="l",xlab=expression(mu[0])
,ylab=expression(Pr(mu<=mu[0])))

## use the cdf to find the 95% credible region.
lcb = cdf$x[with(cdf,which.max(x[y<=0.025]))]
ucb = cdf$x[with(cdf,which.max(x[y<=0.975]))]
cat(paste("Approximate 95% credible interval : ["
,round(lcb,4)," ",round(ucb,4),"]\n",sep=""))

## find the posterior mean, variance and std. deviation
## using Simpson's rule and the output from the previous example
dens = mu*results$posterior # calculate mu*f(mu | x, n)
post.mean = sintegral(mu,dens)$value

dens = (mu-post.mean)^2*results$posterior
post.var = sintegral(mu,dens)$value
post.sd = sqrt(post.var)

# calculate an approximate 95% credible region using the posterior mean and
# std. deviation
lb = post.mean-qnorm(0.975)*post.sd
ub = post.mean+qnorm(0.975)*post.sd

cat(paste("Approximate 95% credible interval : ["
,round(lb,4)," ",round(ub,4),"]\n",sep=""))

# NOTE: All the examples given above can now be done trivially in this package

## find the posterior CDF using the results from the previous example
results = poisgcp(y,"user",mu=mu,mu.prior=mu.prior)
cdf = cdf(results)
```

```
curve(cdf,type="l",xlab=expression(mu[0])
,ylab=expression(Pr(mu<=mu[0])))

## use the quantile function to find the 95% credible region.
ci = quantile(results, c(0.025, 0.975))
cat(paste0("Approximate 95% credible interval : ["
,round(ci[1],4)," ",round(ci[2],4),"]\n"))

## find the posterior mean, variance and std. deviation
## using the output from the previous example
post.mean = mean(results)

post.var = var(results)
post.sd = sd(results)

# calculate an approximate 95% credible region using the posterior mean and
# std. deviation
ci = post.mean + c(-1, 1) * qnorm(0.975) * post.sd

cat(paste("Approximate 95% credible interval : ["
,round(ci[1],4)," ",round(ci[2],4),"]\n",sep=""))

## Example 10.1 Dianna's prior
# Firstly we need to write a function that replicates Diana's prior
f = function(mu){
   result = rep(0, length(mu))
   result[mu >=0 & mu <=2] = mu[mu >=0 & mu <=2]
   result[mu >=2 & mu <=4] = 2
   result[mu >=4 & mu <=8] = 4 - 0.5 * mu[mu >=4 & mu <=8]

   ## we don't need to scale so the prior integrates to one,
   ## but it makes the results nicer to see

   A = 2 + 4 + 4
   result = result / A

   return(result)
 }

 results = poisgcp(y, mu = c(0, 10), mu.prior = f)
```

`print.Bolstad` *Print method for objects of class* `Bolstad`

Description

This function provides a print summary method for the output of `bayes.lm`.

Usage

```
## S3 method for class 'Bolstad'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	an object of class Bolstad
digits	number of digits to print
...	any other arguments that are to be passed to print.default

Details

if x has both class Bolstad and lm then a print method similar to print.lm is called, otherwise print.default is called

Author(s)

James Curran

See Also

bayes.lm

---

print.sintegral *Generic print method*

Description

Print the value of an sintegral object, specifically the (approximate) value of the integral.

Usage

```
## S3 method for class 'sintegral'
print(x, ...)
```

Arguments

x	An object of type sintegral–see sintegral.
...	other parameters passed to paste0.

print.sscsamp *Print method for objects of class* sscsample

Description

This function provides a print summary method for the output of sscsample. The sscsample produces a large number of samples from a fixed population using either simple random, stratified, or cluster sampling. This function provides the means of each sample plus the number of observations from each ethnicity stratum in the sample.

Usage

```
## S3 method for class 'sscsamp'
print(x, ...)
```

Arguments

x	an object of class sscsamp produced by sscsample
...	any other arguments that are to be passed to cat

Author(s)

James Curran

See Also

sscsample

quantile.Bolstad *Posterior quantiles*

Description

Posterior quantiles

Usage

```
## S3 method for class 'Bolstad'
quantile(x, probs = seq(0, 1, 0.25), ...)
```

Arguments

x	an object of class Bolstad
probs	numeric vector of probabilities with values in $[0, 1]$.
...	any extra arguments needed.

Details

If x is of class Bolstad then this will find the quantiles of the posterior distribution using numerical integration and linear interpolation if necessary.

---

sd *Standard deviation generic*

Description

Standard deviation generic

Usage

```
sd(x, ...)
```

Arguments

x	an object.
...	Any additional arguments to be passed to sd.

---

sd.Bolstad *Posterior standard deviation*

Description

Posterior standard deviation

Usage

```
## S3 method for class 'Bolstad'
sd(x, ...)
```

Arguments

x	an object of class Bolstad for which we want to compute the standard deviation.
...	Any additional arguments to be passed to sd.

Calculate the posterior standard deviation of an object of class Bolstad. If the object has a member sd then it will return this value otherwise it will calculate the posterior standard deviation $sd[\theta|x]$ using linear interpolation to approximate the density function and numerical integration where θ is the variable for which we want to do Bayesian inference, and x is the data.

Author(s)

James M. Curran

Examples

```
## The usefulness of this method is really highlighted when we have a general
## continuous prior. In this example we are interested in the posterior
## standard deviation of an normal mean. Our prior is triangular over [-3, 3]
set.seed(123)
x = rnorm(20, -0.5, 1)

mu = seq(-3, 3, by = 0.001)

mu.prior = rep(0, length(mu))
mu.prior[mu <= 0] = 1 / 3 + mu[mu <= 0] / 9
mu.prior[mu > 0] = 1 / 3 - mu[mu > 0] / 9

results = normgcp(x, 1, density = "user", mu = mu, mu.prior = mu.prior, plot = FALSE)
sd(results)
```

---

sintegral *Numerical integration using Simpson's Rule*

Description

Takes a vector of x values and a corresponding set of postive $f(x) = y$ values, or a function, and evaluates the area under the curve:

$$\int f(x)dx$$

.

Usage

```
sintegral(x, fx, n.pts = max(256, length(x)))
```

Arguments

x	a sequence of x values.
fx	the value of the function to be integrated at x or a function
n.pts	the number of points to be used in the integration. If x contains more than n.pts then n.pts will be set to `length(x)`

Value

A list containing two elements, `value` - the value of the intergral, and `cdf` - a list containing elements x and y which give a numeric specification of the cdf.

Examples

```
## integrate the normal density from -3 to 3
x = seq(-3, 3, length = 100)
fx = dnorm(x)
estimate = sintegral(x,fx)$value
true.val = diff(pnorm(c(-3,3)))
abs.error = abs(estimate-true.val)
rel.pct.error =  100*abs(estimate-true.val)/true.val
cat(paste("Absolute error :",round(abs.error,7),"\n"))
cat(paste("Relative percentage error :",round(rel.pct.error,6),"percent\n"))

## repeat the example above using dnorm as function
x = seq(-3, 3, length = 100)
estimate = sintegral(x,dnorm)$value
true.val = diff(pnorm(c(-3,3)))
abs.error = abs(estimate-true.val)
rel.pct.error =  100*abs(estimate-true.val)/true.val
cat(paste("Absolute error :",round(abs.error,7),"\n"))
cat(paste("Relative percentage error :",round(rel.pct.error,6)," percent\n"))

## use the cdf

cdf = sintegral(x,dnorm)$cdf
plot(cdf, type = 'l', col = "black")
lines(x, pnorm(x), col = "red", lty = 2)

## integrate the function x^2-1 over the range 1-2
x = seq(1,2,length = 100)
sintegral(x,function(x){x^2-1})$value

## compare to integrate
integrate(function(x){x^2-1},1,2)
```

---

slug *Slug data*

Description

Lengths and weights of 100 slugs from the species Limax maximus collected around Hamilton, New Zealand.

Usage

```
slug
```

Format

A data frame with 100 observations on the following 4 variables.

- length. length (mm) of the slug
- weight. weight (g) of the slug
- log.len. natural logarithm of the `length`
- log.wt. natural logarithm of the `weight`

References

Barker, G. and McGhie, R. (1984). The Biology of Introduced Slugs (Pulmonata) in New Zealand: Introduction and Notes on Limax Maximus, NZ Entomologist 8, 106–111.

Examples

```
data(slug)
plot(weight~length, data = slug)
plot(log.wt~log.len, data = slug)
```

---

sscsample *Simple, Stratified and Cluster Sampling*

Description

Samples from a fixed population using either simple random sampling, stratitified sampling or cluster sampling.

Usage

```
sscsample(
  size,
  n.samples,
  sample.type = c("simple", "cluster", "stratified"),
  x = NULL,
  strata = NULL,
  cluster = NULL
)
```

Arguments

`size`	the desired size of the sample
`n.samples`	the number of repeat samples to take
`sample.type`	the sampling method. Can be one of "simple", "stratified", "cluser" or 1, 2, 3 where 1 corresponds to "simple", 2 to "stratified" and 3 to "cluster"

x	a vector of measurements for each unit in the population. By default x is not used, and the builtin data set sscsample.data is used
strata	a corresponding vector for each unit in the population indicating membership to a stratum
cluster	a corresponding vector for each unit in the population indicating membership to a cluster

Value

A list will be returned with the following components:

samples	a matrix with the number of rows equal to size and the number of columns equal to n.samples. Each column corresponds to a sample drawn from the population
s.strata	a matrix showing how many units from each stratum were included in the sample
means	a vector containing the mean of each sample drawn

Author(s)

James M. Curran, Dept. of Statistics, University of Auckland. Janko Dietzsch, Proteomics Algorithm and Simulation,Zentrum f. Bioinformatik Tuebingen Fakultaet f. Informations- und Kognitionswissenschaften, Universitaet Tuebingen

Examples

```
## Draw 200 samples of size 20 using simple random sampling
sscsample(20,200)

## Draw 200 samples of size 20 using simple random sampling and store the
## results. Extract the means of all 200 samples, and the 50th sample
res = sscsample(20,200)
res$means
res$samples[,50]
```

---

sscsample.data *Data for simple random sampling, stratified sampling, and clusting sampling experiments*

Description

A simulated population made up of 100 individuals. The individuals come from three ethnic groups with population proportions of 40%, 40%, and 20%, respectively. There are twenty neighborhoods, and five individuals live in each one. Now, the income distribution may be different for the three ethnic groups. Also, individuals in the same neighborhood tend to be more similar than individuals in different neighborhoods.

Usage

```
sscsample.data
```

Format

A data frame with 100 observations on the following 3 variables.

- income. Simulated income in $10,000
- ethnicity. A numerical vector indicating the ethnic group of the observation
- neighborhood. A numeric vector indicating the neighborhood of the observation

Examples

```
data(sscsample.data)
plot(income~ethnicity, data = sscsample.data)
```

---

summary.Bolstad *Summarizing Bayesian Multiple Linear Regression*

Description

summary method for output of bayes.lm.

Usage

```
## S3 method for class 'Bolstad'
summary(object, ...)
```

Arguments

object	an object of "Bolstad" that is the result of a call to bayes.lm
...	any further arguments to be passed to print

See Also

The function to fit the model bayes.lm

The function coef to extract the matrix of posterior means along with standard errors and t-statistics.

var *Variance generic*

Description

Variance generic

Usage

```
var(x, ...)
```

Arguments

x	an object for which we want to compute the variance
...	Any additional arguments to be passed to var.

xdesign *Monte Carlo study of randomized and blocked designs*

Description

Simulates completely randomized design and randomized block designs from a population of experimental units with underlying response values y and underlying other variable values x (possibly lurking)

Usage

```
xdesign(
  x = NULL,
  y = NULL,
  corr = 0.8,
  size = 20,
  n.treatments = 4,
  n.rep = 500,
  ...
)
```

Arguments

x	a set of lurking values which are correlated with the response
y	a set of response values
corr	the correlation between the response and lurking variable
size	the size of the treatment groups
n.treatments	the number of treatments
n.rep	the number of Monte Carlo replicates
...	additional parameters which are passed to Bolstad.control

Value

If the ouput of xdesign is assigned to a variable, then a list is returned with the following components:

`block.means`	a vector of the means of the lurking variable from each replicate of the simulation stored by treatment number within replicate number
`treat.means`	a vector of the means of the response variable from each replicate of the simulation stored by treatment number within replicate number
`ind`	a vector containing the treatment group numbers. Note that there will be twice as many group numbers as there are treatments corresponding to the simulations done using a completely randomized design and the simulations done using a randomized block design

Examples

```
# Carry out simulations using the default parameters

xdesign()

# Carry out simulations using a simulated response with 5 treaments,
# groups of size 25, and a correlation of -0.6 between the response
# and lurking variable

xdesign(corr = -0.6, size = 25, n.treatments = 5)
```

Index