Package: BAS (via r-universe)

October 11, 2026

Version 2.0.2

Title Bayesian Variable Selection and Model Averaging using Bayesian Adaptive Sampling

Depends R (>= 3.0)

Imports stats, graphics, utils, grDevices

Suggests MASS, knitr, ggplot2, GGally, rmarkdown, roxygen2, dplyr, glmbb, testthat, covr, faraway

Description Package for Bayesian Variable Selection and Model Averaging in linear models and generalized linear models using stochastic or deterministic sampling without replacement from posterior distributions. Prior distributions on coefficients are from Zellner's g-prior or mixtures of g-priors corresponding to the Zellner-Siow Cauchy Priors or the mixture of g-priors from Liang et al (2008) <DOI:10.1198/016214507000001337> for linear models or mixtures of g-priors from Li and Clyde (2019) <DOI:10.1080/01621459.2018.1469992> in generalized linear models. Other model selection criteria include AIC, BIC and Empirical Bayes estimates of g. Sampling probabilities may be updated based on the sampled models using sampling w/out replacement or an efficient MCMC algorithm which samples models using a tree structure of the model space as an efficient hash table. See Clyde, Ghosh and Littman (2010) <DOI:10.1198/jcgs.2010.09049> for details on the sampling algorithms. Uniform priors over all models or beta-binomial prior distributions on model size are allowed, and for large p truncated priors on the model space may be used to enforce sampling models that are full rank. The user may force variables to always be included in addition to imposing constraints that higher order interactions are included only if their parents are included in the model. This material is based upon work supported by the National Science Foundation under Division of Mathematical Sciences grant 1106891. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily

reflect the views of the National Science Foundation.

License GPL (>= 3)

URL https://merliseclyde.github.io/BAS/,
https://github.com/merliseclyde/BAS

BugReports https://github.com/merliseclyde/BAS/issues

NeedsCompilation yes

ByteCompile yes

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.3

Author Merlise Clyde [aut, cre, cph] (ORCID=0000-0002-3595-1872),
Michael Littman [ctb], Joyee Ghosh [ctb], Yingbo Li [ctb],
Betsy Bersson [ctb], Don van de Bergh [ctb], Quanli Wang [ctb]

Maintainer Merlise Clyde <clyde@duke.edu>

Repository https://cran.r-universe.dev

Date/Publication 2025-12-23 06:10:56 UTC

RemoteUrl https://github.com/cran/BAS

RemoteRef HEAD

RemoteSha 25f023b4a318777cb00011b954626dde1d71db4c

Contents

BAS . . . 3
bas.glm . . . 5
bas.lm . . . 10
Bayes.outlier . . . 20
bayesglm.fit . . . 21
Bernoulli . . . 23
Bernoulli.heredity . . . 24
beta.binomial . . . 24
beta.prime . . . 25
bodyfat . . . 26
CCH . . . 28
climate . . . 29
coef.bas . . . 30
confint.coef.bas . . . 32
confint.pred.bas . . . 33
cv.summary.bas . . . 34
diagnostics . . . 35
EB.global . . . 36
EB.local . . . 38
eplogprob . . . 38

eplogprob.marg . . . 40
fitted.bas . . . 41
force.heredity.bas . . . 43
g.prior . . . 44
Hald . . . 45
hyper.g . . . 45
hyper.g.n . . . 46
hypergeometric1F1 . . . 47
hypergeometric2F1 . . . 48
IC.prior . . . 49
image.bas . . . 50
intrinsic . . . 52
Jeffreys . . . 53
list2matrix.bas . . . 54
list2matrix.which . . . 55
phi1 . . . 56
plot.bas . . . 57
plot.coef.bas . . . 59
plot.confint.bas . . . 60
predict.bas . . . 62
predict.basglm . . . 64
print.bas . . . 66
protein . . . 67
robust . . . 68
summary.bas . . . 69
tCCH . . . 70
testBF.prior . . . 71
TG . . . 72
tr.beta.binomial . . . 73
tr.poisson . . . 74
tr.power.prior . . . 75
trCCH . . . 76
uniform . . . 77
update.bas . . . 78
variable.names.pred.bas . . . 79
which.matrix . . . 80

Index **82**

Description

Implementation of Bayesian Model Averaging in linear models using stochastic or deterministic sampling without replacement from posterior distributions. Prior distributions on coefficients are of the form of Zellner's g-prior or mixtures of g-priors. Options include the Zellner-Siow Cauchy Priors, the Liang et al hyper-g priors, Local and Global Empirical Bayes estimates of g, and other default model selection criteria such as AIC and BIC. Sampling probabilities may be updated based on the sampled models.

Details

_PACKAGE

Author(s)

Merlise Clyde,
Maintainer: Merlise Clyde <clyde@stat.duke.edu>

References

Clyde, M. Ghosh, J. and Littman, M. (2010) Bayesian Adaptive Sampling for Variable Selection and Model Averaging. Journal of Computational Graphics and Statistics. 20:80-101 doi:10.1198/jcgs.2010.09049

Clyde, M. and George, E. I. (2004) Model uncertainty. Statist. Sci., 19, 81-94. doi:10.1214/088342304000000035

Clyde, M. (1999) Bayesian Model Averaging and Model Search Strategies (with discussion). In Bayesian Statistics 6. J.M. Bernardo, A.P. Dawid, J.O. Berger, and A.F.M. Smith eds. Oxford University Press, pages 157-185.

Li, Y. and Clyde, M. (2018) Mixtures of g-priors in Generalized Linear Models. Journal of the American Statistical Association, 113:524, 1828-1845 doi:10.1080/01621459.2018.1469992

Liang, F., Paulo, R., Molina, G., Clyde, M. and Berger, J.O. (2008) Mixtures of g-priors for Bayesian Variable Selection. Journal of the American Statistical Association. 103:410-423.

doi:10.1198/016214507000001337

See Also

bas.lm bas.glm

Other bas methods: bas.lm(), coef.bas(), confint.coef.bas(), confint.pred.bas(), diagnostics(), fitted.bas(), force.heredity.bas(), image.bas(), plot.confint.bas(), predict.bas(), predict.basglm(), summary.bas(), update.bas(), variable.names.pred.bas()

Examples

```
data("Hald")
hald.gprior =  bas.lm(Y ~ ., data=Hald, alpha=13, prior="g-prior")

# more complete demos
```

```
demo(BAS.hald)
## Not run:
demo(BAS.USCrime)

## End(Not run)
```

`bas.glm` *Bayesian Adaptive Sampling Without Replacement for Variable Selection in Generalized Linear Models*

Description

Sample with or without replacement from a posterior distribution on GLMs

Usage

```
bas.glm(
  formula,
  family = binomial(link = "logit"),
  data,
  weights,
  subset,
  contrasts = NULL,
  offset,
  na.action = "na.omit",
  n.models = NULL,
  betaprior = CCH(alpha = 0.5, beta = as.numeric(nrow(data)), s = 0),
  modelprior = beta.binomial(1, 1),
  initprobs = "Uniform",
  include.always = ~1,
  method = "MCMC",
  update = NULL,
  bestmodel = NULL,
  prob.rw = 0.5,
  burnin.iterations = NULL,
  MCMC.iterations = NULL,
  thin = 1,
  control = glm.control(),
  laplace = FALSE,
  renormalize = FALSE,
  force.heredity = FALSE,
  GROW = TRUE,
  expand = 1.25,
  n.models.init = 2500,
  bigmem = FALSE
)
```

Arguments

formula	generalized linear model formula for the full model with all predictors, Y ~ X. All code assumes that an intercept will be included in each model.
family	a description of the error distribution and link function for exponential family; currently only 'binomial()' with the logistic link and 'poisson()' and 'Gamma()'with the log link are available.
data	data frame
weights	optional vector of weights to be used in the fitting process. May be missing in which case weights are 1.
subset	subset of data used in fitting
contrasts	an optional list. See the contrasts.arg of 'model.matrix.default()'.
offset	a priori known component to be included in the linear predictor; by default 0.
na.action	a function which indicates what should happen when the data contain NAs. The default is "na.omit".
n.models	number of unique models to keep. If NULL, BAS will attempt to enumerate unless p > 35 or method="MCMC". For any of methods using MCMC algorithms that sample with replacement, sampling will stop when the number of iterations exceeds 'MCMC.iterations'. On exit 'n.models' is updated to reflect the unique number of models that have been sampled.
betaprior	Prior on coefficients for model coefficients (except intercept). Options include g.prior, CCH, robust, intrinsic, beta.prime, EB.local, AIC, and BIC.
modelprior	Family of prior distribution on the models. Choices include uniform, Bernoulli, beta.binomial, truncated Beta-Binomial, tr.beta.binomial, and truncated power family tr.power.prior.
initprobs	vector of length p with the initial inclusion probabilities used for sampling without replacement (the intercept will be included with probability one and does not need to be added here) or a character string giving the method used to construct the sampling probabilities if "Uniform" each predictor variable is equally likely to be sampled (equivalent to random sampling without replacement). If "eplogp", use the eplogprob function to approximate the Bayes factor using p-values to find initial marginal inclusion probabilities and sample without replacement using these inclusion probabilities, which may be updated using estimates of the marginal inclusion probabilities. "eplogp" assumes that MLEs from the full model exist; for problems where that is not the case or 'p' is large, initial sampling probabilities may be obtained using eplogprob.marg which fits a model to each predictor separately. To run a Markov Chain to provide initial estimates of marginal inclusion probabilities, use method="MCMC+BAS" below. While the initprobs are not used in sampling for method="MCMC", this determines the order of the variables in the lookup table and affects memory allocation in large problems where enumeration is not feasible. For variables that should always be included set the corresponding initprobs to 1, to override the 'modelprior' or use 'include.always' to force these variables to always be included in the model.

include.always A formula with terms that should always be included in the model with probability one. By default this is '~ 1' meaning that the intercept is always included. This will also override any of the values in 'initprobs' above by setting them to 1.

method A character variable indicating which sampling method to use: method="BAS" uses Bayesian Adaptive Sampling (without replacement) using the sampling probabilities given in initprobs and updates using the marginal inclusion probabilities to direct the search/sample; method="MCMC" combines a random walk Metropolis Hastings (as in MC3 of Raftery et al 1997) with a random swap of a variable included with a variable that is currently excluded (see Clyde, Ghosh, and Littman (2010) for details); method="MCMC+BAS" runs an initial MCMC as above to calculate marginal inclusion probabilities and then samples without replacement as in BAS; method = "deterministic" runs a deterministic sampler using the initial probabilities (no updating); this is recommended for fast enumeration or if a model of independence is a good approximation to the joint posterior distribution of the model indicators. For BAS, the sampling probabilities can be updated as more models are sampled. (see 'update' below). We recommend "MCMC+BAS" or "MCMC" for high dimensional problems.

update number of iterations between potential updates of the sampling probabilities in the "BAS" method. If NULL do not update, otherwise the algorithm will update using the marginal inclusion probabilities as they change while sampling takes place. For large model spaces, updating is recommended. If the model space will be enumerated, leave at the default.

bestmodel optional binary vector representing a model to initialize the sampling. If NULL sampling starts with the null model

prob.rw For any of the MCMC methods, probability of using the random-walk proposal; otherwise use a random "flip" move to propose a new model.

burnin.iterations
Number of iterations to discard as part of burnin when using any of the MCMC options; should be greater than 'n.models'. By default 10*p.

MCMC.iterations
Number of MCMC iterations for sampling using any of the MCMC options; should be greater than 'n.models'. By default 1000*p.

thin For "MCMC", thin the MCMC chain every "thin" iterations; default is no thinning. For large p, thinning can be used to significantly reduce memory requirements as models and associated summaries are saved only every thin iterations. For thin = p, the model and associated output are recorded every p iterations,similar to the Gibbs sampler in SSVS.

control a list of parameters that control convergence in the fitting process. See the documentation for `glm.control()`

laplace logical variable for whether to use a Laplace approximate for integration with respect to g to obtain the marginal likelihood. If FALSE the Cephes library is used which may be inaccurate for large n or large values of the Wald Chisquared statistic.

renormalize logical variable for whether posterior probabilities should be based on renormalizing marginal likelihoods times prior probabilities or use Monte Carlo frequencies. Applies only to MCMC sampling.

force.heredity	Logical variable to force all levels of a factor to be included together and to include higher order interactions only if lower order terms are included. Currently only supported with 'method='MCMC'' and 'method='BAS'' (experimental). Default is FALSE.
GROW	Logical variable to indicate that the output vectors in MCMC are growable. Rather than allocate space based on 'n.models', the vectors will grow as needed if the number of unique models sampled exceeds the initial size of the allocated output vectors controlled by 'n.models.init'. This is useful when 'n.models' is unknown before reaching 'MCMC.iterations'. Default is TRUE.
expand	variable to control how much to grow vectors with GROW = TRUE if number of unique models exceeds the current size of the vectors. The default is 1.25 times, which allows vectors to grow by 25 percent.
n.models.init	Initial size of output vectors if GROW = TRUE. The default is 'n.models = 2500'.
bigmem	Logical variable to indicate that there is access to large amounts of memory (physical or virtual) for enumeration with large model spaces, e.g. > 2^25.

Details

BAS provides several search algorithms to find high probability models for use in Bayesian Model Averaging or Bayesian model selection. For p less than 20-25, BAS can enumerate all models depending on memory availability, for larger p, BAS samples without replacement using random or deterministic sampling. The Bayesian Adaptive Sampling algorithm of Clyde, Ghosh, Littman (2010) samples models without replacement using the initial sampling probabilities, and will optionally update the sampling probabilities every "update" models using the estimated marginal inclusion probabilities. BAS uses different methods to obtain the `initprobs`, which may impact the results in high-dimensional problems. The deterministic sampler provides a list of the top models in order of an approximation of independence using the provided `initprobs`. This may be effective after running the other algorithms to identify high probability models and works well if the correlations of variables are small to modest. The priors on coefficients are mixtures of g-priors that provide approximations to the power prior.

Value

`bas.glm` returns an object of class `basglm`

An object of class `basglm` is a list containing at least the following components:

postprobs	the posterior probabilities of the models selected
priorprobs	the prior probabilities of the models selected
logmarg	values of the log of the marginal likelihood for the models
n.vars	total number of independent variables in the full model, including the intercept
size	the number of independent variables in each of the models, includes the intercept
which	a list of lists with one list per model with variables that are included in the model
probne0	the posterior probability that each variable is non-zero
mle	list of lists with one list per model giving the GLM estimate of each (nonzero) coefficient for each model.

mle.se	list of lists with one list per model giving the GLM standard error of each coefficient for each model
deviance	the GLM deviance for each model
modelprior	the prior distribution on models that created the BMA object
Q	the Q statistic for each model used in the marginal likelihood approximation
Y	response
X	matrix of predictors
family	family object from the original call
betaprior	family object for prior on coefficients, including hyperparameters
modelprior	family object for prior on the models
include.always	indices of variables that are forced into the model

Author(s)

Merlise Clyde (<clyde@duke.edu>), Quanli Wang and Yingbo Li

References

Li, Y. and Clyde, M. (2018) Mixtures of g-priors in Generalized Linear Models. Journal of the American Statistical Association. 113:1828-1845
doi:10.1080/01621459.2018.1469992
Clyde, M. Ghosh, J. and Littman, M. (2010) Bayesian Adaptive Sampling for Variable Selection and Model Averaging. Journal of Computational Graphics and Statistics. 20:80-101
doi:10.1198/jcgs.2010.09049
Raftery, A.E, Madigan, D. and Hoeting, J.A. (1997) Bayesian Model Averaging for Linear Regression Models. Journal of the American Statistical Association.

Examples

```
library(MASS)
data(Pima.tr)


# enumeration  with default method="BAS"
pima.cch = bas.glm(type ~ ., data=Pima.tr, n.models= 2^7,
              method="BAS",
              betaprior=CCH(a=1, b=532/2, s=0), family=binomial(),
              modelprior=beta.binomial(1,1))

summary(pima.cch)
image(pima.cch)

# Note MCMC.iterations are set to 2500 for illustration purposes due to time
# limitations for running examples on CRAN servers.
# Please check convergence diagnostics and run longer in practice

pima.robust = bas.glm(type ~ ., data=Pima.tr, n.models= 2^7,
              method="MCMC", MCMC.iterations=2500,
```

```
              betaprior=robust(), family=binomial(),
              modelprior=beta.binomial(1,1))

pima.BIC = bas.glm(type ~ ., data=Pima.tr, n.models= 2^7,
              method="MCMC+BAS", MCMC.iterations=2500,
              betaprior=bic.prior(), family=binomial(),
              modelprior=uniform())
# Poisson example
if(requireNamespace("glmbb", quietly=TRUE)) {
  data(crabs, package='glmbb')
  #short run for illustration
  crabs.bas = bas.glm(satell ~ color*spine*width + weight, data=crabs,
                      family=poisson(),
                      betaprior=EB.local(), modelprior=uniform(),
                      method="MCMC", n.models=2^10, MCMC.iterations=2500,
                      prob.rw=.95)

 # Gamma example
 if(requireNamespace("faraway", quietly=TRUE)) {
    data(wafer, package='faraway')

    wafer_bas = bas.glm(resist~ ., data=wafer,  include.always = ~ .,
                        betaprior = bic.prior() ,
                        family = Gamma(link = "log"))
  }
}
```

`bas.lm` *Bayesian Adaptive Sampling for Bayesian Model Averaging and Variable Selection in Linear Models*

Description

Sample without replacement from a posterior distribution on models

Usage

```
bas.lm(
  formula,
  data,
  subset,
  weights,
  contrasts = NULL,
  na.action = "na.omit",
  n.models = NULL,
  prior = "ZS-null",
  alpha = NULL,
  modelprior = beta.binomial(1, 1),
  initprobs = "Uniform",
```

```
  include.always = ~1,
  method = "BAS",
  update = NULL,
  bestmodel = NULL,
  prob.local = 0,
  prob.rw = 0.5,
  burnin.iterations = NULL,
  MCMC.iterations = NULL,
  lambda = NULL,
  delta = 0.025,
  thin = 1,
  renormalize = FALSE,
  importance.sampling = FALSE,
  FPS = "none",
  force.heredity = FALSE,
  pivot = TRUE,
  tol = 1e-07,
  GROW = TRUE,
  expand = 1.05,
  n.models.init = 2500,
  bigmem = FALSE
)
```

Arguments

formula	linear model formula for the full model with all predictors, Y ~ X. All code assumes that an intercept will be included in each model and that the X's will be centered.
data	a data frame. Factors will be converted to numerical vectors based on the using 'model.matrix'.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
weights	an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector. If non-NULL, Bayes estimates are obtained assuming that $Y_i \sim N(x_i^T\beta, \sigma^2/w_i)$.
contrasts	an optional list. See the contrasts.arg of 'model.matrix.default()'.
na.action	a function which indicates what should happen when the data contain NAs. The default is "na.omit".
n.models	number of models to sample either without replacement (method="BAS" or "MCMC+BAS") or initial number of models to sample with replacement (method="MCMC" or "AMCMC"). If NULL,BAS with method="BAS" will try to enumerate/sample the min(2^p, 2^16). If 'n.models' > 2^25, the user should use 'bigmem = TRUE' to sample/enumerate 'n.models'. With method="MCMC" or "AMCMC", 'n.models' controls the initial number of models with the default for n.models = min(2000, 2^p). Sampling will stop once burnin.iterations + MCMC.iterations are exceeded and n.models will be increased/decreased as needed to store the unique models sampled. On exit 'n.models' is the number of unique models that have

been sampled. For sampling with replacement (MCMC or AMCMC) the counts for the number of times a models is sampled is stored in the output as "freq".

`prior` prior distribution for regression coefficients. Choices include

- "AIC"
- "BIC"
- "g-prior", Zellner's g prior where 'g' is specified using the argument 'alpha'
- "JZS" Jeffreys-Zellner-Siow prior which uses the Jeffreys prior on sigma and the Zellner-Siow Cauchy prior on the coefficients. The optional parameter 'alpha' can be used to control the squared scale of the prior, where the default is alpha=1. Setting 'alpha' is equal to rscale^2 in the BayesFactor package of Morey. This uses QUADMATH for numerical integration of g.
- "ZS-null", a Laplace approximation to the 'JZS' prior for integration of g. alpha = 1 only. We recommend using 'JZS' for accuracy and compatibility with the BayesFactor package, although it is slower.
- "ZS-full" (to be deprecated)
- "hyper-g", a mixture of g-priors where the prior on g/(1+g) is a Beta(1, alpha/2) as in Liang et al (2008). This uses the Cephes library for evaluation of the marginal likelihoods and may be numerically unstable for large n or R2 close to 1. Default choice of alpha is 3.
- "hyper-g-laplace", Same as above but using a Laplace approximation to integrate over the prior on g.
- "hyper-g-n", a mixture of g-priors that where u = g/n and u ~ Beta(1, alpha/2) to provide consistency when the null model is true.
- "EB-local", use the MLE of g from the marginal likelihood within each model
- "EB-global" uses an EM algorithm to find a common or global estimate of g, averaged over all models. When it is not possible to enumerate all models, the EM algorithm uses only the models sampled under EB-local.

`alpha` optional hyperparameter in g-prior or hyper g-prior. For Zellner's g-prior, alpha = g, for the Liang et al hyper-g or hyper-g-n method, recommended choice is alpha are between (2 < alpha < 4), with alpha = 3 the default. For the Zellner-Siow prior alpha = 1 by default, but can be used to modify the rate parameter in the gamma prior on g,

$$1/g \sim G(1/2, n * \alpha/2)$$

so that

$$\beta \sim C(0, \sigma^2 \alpha (X'X/n)^{-1})$$

. If alpha = NULL, then the following defaults are used currently:

- "g-prior" = n,
- "hyper-g" = 3,
- "EB-local" = 2,
- "BIC" = n,
- "ZS-null" = 1,
- "ZS-full" = n,
- "hyper-g-laplace" = 3,

- "AIC" = 0,
- "EB-global" = 2,
- "hyper-g-n" = 3,
- "JZS" = 1,

Note that Porwal & Raftery (2022) recommend alpha = sqrt(n) for the g-prior based on extensive range of simulations and examples for comparing BMA.

`modelprior` A function for a family of prior distribution on the models. Choices include `uniform` `Bernoulli` or `beta.binomial`, `tr.beta.binomial`, (with truncation) `tr.poisson` (a truncated Poisson), and `tr.power.prior` (a truncated power family), with the default being a `beta.binomial(1,1)`. Truncated versions are useful for p > n.

`initprobs` Vector of length p or a character string specifying which method is used to create the vector. This is used to order variables for sampling all methods for potentially more efficient storage while sampling and provides the initial inclusion probabilities used for sampling without replacement with method="BAS". Options for the character string giving the method are: "Uniform" or "uniform" where each predictor variable is equally likely to be sampled (equivalent to random sampling without replacement); "eplogp" uses the `eplogprob` function to approximate the Bayes factor from p-values from the full model to find initial marginal inclusion probabilities; "marg-eplogp" uses`eplogprob.marg` function to approximate the Bayes factor from p-values from the full model each simple linear regression. To run a Markov Chain to provide initial estimates of marginal inclusion probabilities for "BAS", use method="MCMC+BAS" below. While the initprobs are not used in sampling for method="MCMC", this determines the order of the variables in the lookup table and affects memory allocation in large problems where enumeration is not feasible. For variables that should always be included set the corresponding initprobs to 1, to override the 'modelprior' or use 'include.always' to force these variables to always be included in the model.

`include.always` A formula with terms that should always be included in the model with probability one. By default this is '~ 1' meaning that the intercept is always included. This will also override any of the values in 'initprobs' above by setting them to 1.

`method` A character variable indicating which sampling method to use:

- "deterministic" uses the "top k" algorithm described in Ghosh and Clyde (2011) to sample models in order of approximate probability under conditional independence using the "initprobs". This is the most efficient algorithm for enumeration.
- "BAS" uses Bayesian Adaptive Sampling (without replacement) using the sampling probabilities given in initprobs under a model of conditional independence. These can be updated based on estimates of the marginal inclusion probabilities.
- "MCMC" samples with replacement via a MCMC algorithm that combines the birth/death random walk in Hoeting et al (1997) of MC3 with a random swap move to interchange a variable in the model with one currently excluded as described in Clyde, Ghosh and Littman (2010).

- "MCMC+BAS" runs an initial MCMC to calculate marginal inclusion probabilities and then samples without replacement as in BAS. For BAS, the sampling probabilities can be updated as more models are sampled. (see update below).
- "AMCMC" uses an adaptive proposal based on factoring the proposal distribution as a product conditional probabilities estimated from the past draws. If 'importance.sampling = FALSE' this uses an adaptive independent Metropolis-Hasting algorithm, with if 'importance.sampling = TRUE' uses importance sampling combined with Horwitz-Thompson estimates of posterior model and inclusion probabilities.

`update` number of iterations between potential updates of the sampling probabilities for method "BAS" or "MCMC+BAS". If NULL do not update, otherwise the algorithm will update using the marginal inclusion probabilities as they change while sampling takes place. For large model spaces, updating is recommended. If the model space will be enumerated, leave at the default.

`bestmodel` optional binary vector representing a model to initialize the sampling. If NULL sampling starts with the null model

`prob.local` A future option to allow sampling of models "near" the median probability model. Not used at this time.

`prob.rw` For any of the MCMC methods, probability of using the random-walk Metropolis proposal; otherwise use a random "flip" move to propose swap a variable that is excluded with a variable in the model.

`burnin.iterations`
Number of burnin iterations for the MCMC sampler; the default is p*25 if not set by the user.

`MCMC.iterations`
Number of iterations for the MCMC sampler; the default is p*1000 if not set by the user.

`lambda` Parameter in the AMCMC algorithm to insure positive definite covariance of gammas for adaptive conditional probabilities prior based on prior degrees of freedom pseudo in Inverse-Wishart. Default is set to p + 2.

`delta` truncation parameter to prevent sampling probabilities to degenerate to 0 or 1 prior to enumeration for sampling without replacement.

`thin` For "MCMC" or "MCMC+BAS", thin the MCMC chain every "thin" iterations; default is no thinning. For large p, thinning can be used to significantly reduce memory requirements as models and associated summaries are saved only every thin iterations. For thin = p, the model and associated output are recorded every p iterations, similar to the Gibbs sampler in SSVS.

`renormalize` For MCMC sampling, should posterior probabilities be based on renormalizing the marginal likelihoods times prior probabilities (TRUE) or frequencies from MCMC. The latter are unbiased in long runs, while the former may have less variability. May be compared via the diagnostic plot function `diagnostics`. See details in Clyde and Ghosh (2012).

`importance.sampling`
whether to use importance sampling or an independent Metropolis-Hastings algorithm with sampling method="AMCMC" (see above).

FPS — Finite Population Sampling estimator for use with method="MCMC+BAS". Options include "none" (default) which uses the sum of marginal likelihoods times priors for sampled models to estimate the normalizing constant, or "Bayes_HT" which uses an additional correction to the normalizing constant to account for unsampled models using sampling probabilities.

force.heredity — Logical variable to force all levels of a factor to be included together and to include higher order interactions only if lower order terms are included. Currently supported with 'method='MCMC'' and experimentally with 'method='BAS'' on non-Solaris platforms. This is not compatible currently for enforcing hierachical constraints with orthogonal polynomials, poly(x, degree = 3). Without hereditary constraints the number of possible models with all possible interactions is 2^2^k where k is the number of factors with more than 2 levels. With hereditary constraints the number of models is much less, but computing this for k can be quite expensive for large k. For the model y ~ x1*x2*x3*x4*x5*x6) there are 7828353 models, which is more than 2^22. With n.models given, this will limit the number of models to the min(n.models, # models under the heredity constraints) Default is FALSE currently.

pivot — Logical variable to allow pivoting of columns when obtaining the OLS estimates of a model so that models that are not full rank can be fit. Defaults to TRUE. Currently coefficients that are not estimable are set to zero. Use caution with interpreting BMA estimates of parameters.

tol — 1e-7 as

GROW — Logical variable to indicate that the output vectors in MCMC are growable. Rather than allocate space based on 'n.models', the vectors will grow as needed if the number of unique models sampled exceeds the initial size of the allocated output vectors controlled by 'n.models.init'. This is useful when 'n.models' is unknown before reaching 'MCMC.iterations'. Default is TRUE.

expand — variable to control how much to grow vectors with GROW = TRUE if number of unique models exceeds the current size of the vectors. The default is 1.25 times, which allows vectors to grow by 25 percent.

n.models.init — Initial size of output vectors if GROW = TRUE. The default is 'n.models = 2500'.

bigmem — Logical variable to indicate that there is access to large amounts of memory (physical or virtual) for enumeration with large model spaces, e.g. > 2^25. default; used in determining rank of X^TX in Cholesky decomposition with pivoting.

Details

BAS provides several algorithms to sample from posterior distributions of models for use in Bayesian Model Averaging or Bayesian variable selection. For p less than 20-25, BAS can enumerate all models depending on memory availability. As BAS saves all models, MLEs, standard errors, log marginal likelihoods, prior and posterior and probabilities memory requirements grow linearly with M*p where M is the number of models and p is the number of predictors. For example, enumeration with p=21 with 2,097,152 takes just under 2 Gigabytes on a 64 bit machine to store all summaries that would be needed for model averaging. (A future version will likely include an option to not store all summaries if users do not plan on using model averaging or model selection on

Best Predictive models.) For larger p, BAS samples without replacement using random or deterministic sampling. The Bayesian Adaptive Sampling algorithm of Clyde, Ghosh, Littman (2010) samples models without replacement using the initial sampling probabilities, and will optionally update the sampling probabilities every "update" models using the estimated marginal inclusion probabilities. BAS uses different methods to obtain the `initprobs`, which may impact the results in high-dimensional problems. The deterministic sampler provides a list of the top models in order of an approximation of independence using the provided `initprobs`. This may be effective after running the other algorithms to identify high probability models and works well if the correlations of variables are small to modest. We recommend "MCMC" for problems where enumeration is not feasible (memory or time constrained) or even modest p if the number of models sampled is not close to the number of possible models and/or there are significant correlations among the predictors as the bias in estimates of inclusion probabilities from "BAS" or "MCMC+BAS" may be large relative to the reduced variability from using the normalized model probabilities as shown in Clyde and Ghosh, 2012. Diagnostic plots with MCMC can be used to assess convergence. For large problems we recommend thinning with MCMC to reduce memory requirements. The priors on coefficients include Zellner's g-prior, the Hyper-g prior (Liang et al 2008, the Zellner-Siow Cauchy prior, Empirical Bayes (local and global) g-priors. AIC and BIC are also included, while a range of priors on the model space are available.

Value

`bas` returns an object of class `bas`

An object of class `BAS` is a list containing at least the following components:

`postprob`	the posterior probabilities of the models selected
`priorprobs`	the prior probabilities of the models selected
`namesx`	the names of the variables
`R2`	R2 values for the models
`logmarg`	values of the log of the marginal likelihood for the models. This is equivalent to the log Bayes Factor for comparing each model to a base model with intercept only.
`n.vars`	total number of independent variables in the full model, including the intercept
`size`	the number of independent variables in each of the models, includes the intercept
`rank`	the rank of the design matrix; if 'pivot = FALSE', this is the same as size as no checking of rank is conducted.
`which`	a list of lists with one list per model with variables that are included in the model
`probne0`	the posterior probability that each variable is non-zero computed using the renormalized marginal likelihoods of sampled models. This may be biased if the number of sampled models is much smaller than the total number of models. Unbiased estimates may be obtained using method "MCMC".
`mle`	list of lists with one list per model giving the MLE (OLS) estimate of each (nonzero) coefficient for each model. NOTE: The intercept is the mean of Y as each column of X has been centered by subtracting its mean.
`mle.se`	list of lists with one list per model giving the MLE (OLS) standard error of each coefficient for each model

prior	the name of the prior that created the BMA object
alpha	value of hyperparameter in coefficient prior used to create the BMA object.
modelprior	the prior distribution on models that created the BMA object
Y	response
X	matrix of predictors
mean.x	vector of means for each column of X (used in predict.bas)
include.always	indices of variables that are forced into the model

The function summary.bas, is used to print a summary of the results. The function plot.bas is used to plot posterior distributions for the coefficients and image.bas provides an image of the distribution over models. Posterior summaries of coefficients can be extracted using coefficients.bas. Fitted values and predictions can be obtained using the S3 functions fitted.bas and predict.bas. BAS objects may be updated to use a different prior (without rerunning the sampler) using the function update.bas. For MCMC sampling diagnostics can be used to assess whether the MCMC has run long enough so that the posterior probabilities are stable. For more details see the associated demos and vignette.

Author(s)

Merlise Clyde (<clyde@duke.edu>) and Michael Littman

References

Clyde, M. Ghosh, J. and Littman, M. (2010) Bayesian Adaptive Sampling for Variable Selection and Model Averaging. Journal of Computational Graphics and Statistics. 20:80-101 doi:10.1198/jcgs.2010.09049

Clyde, M. and Ghosh. J. (2012) Finite population estimators in stochastic search variable selection. Biometrika, 99 (4), 981-988. doi:10.1093/biomet/ass040

Clyde, M. and George, E. I. (2004) Model Uncertainty. Statist. Sci., 19, 81-94. doi:10.1214/088342304000000035

Clyde, M. (1999) Bayesian Model Averaging and Model Search Strategies (with discussion). In Bayesian Statistics 6. J.M. Bernardo, A.P. Dawid, J.O. Berger, and A.F.M. Smith eds. Oxford University Press, pages 157-185.

Hoeting, J. A., Madigan, D., Raftery, A. E. and Volinsky, C. T. (1999) Bayesian model averaging: a tutorial (with discussion). Statist. Sci., 14, 382-401. doi:10.1214/ss/1009212519

Liang, F., Paulo, R., Molina, G., Clyde, M. and Berger, J.O. (2008) Mixtures of g-priors for Bayesian Variable Selection. Journal of the American Statistical Association. 103:410-423. doi:10.1198/016214507000001337

Porwal, A. and Raftery, A. E. (2022) Comparing methods for statistical inference with model uncertainty PNAS 119 (16) e2120737119 doi:10.1073/pnas.2120737119

Zellner, A. (1986) On assessing prior distributions and Bayesian regression analysis with g-prior distributions. In Bayesian Inference and Decision Techniques: Essays in Honor of Bruno de Finetti, pp. 233-243. North-Holland/Elsevier.

Zellner, A. and Siow, A. (1980) Posterior odds ratios for selected regression hypotheses. In Bayesian Statistics: Proceedings of the First International Meeting held in Valencia (Spain), pp. 585-603.

Rouder, J. N., Speckman, P. L., Sun, D., Morey, R. D., and Iverson, G. (2009). Bayesian t-tests for accepting and rejecting the null hypothesis. Psychonomic Bulletin & Review, 16, 225-237

Rouder, J. N., Morey, R. D., Speckman, P. L., Province, J. M., (2012) Default Bayes Factors for ANOVA Designs. Journal of Mathematical Psychology. 56. p. 356-374.

See Also

summary.bas, coefficients.bas, print.bas, predict.bas, fitted.bas plot.bas, image.bas, eplogprob, update.bas

Other bas methods: BAS, coef.bas(), confint.coef.bas(), confint.pred.bas(), diagnostics(), fitted.bas(), force.heredity.bas(), image.bas(), plot.confint.bas(), predict.bas(), predict.basglm(), summary.bas(), update.bas(), variable.names.pred.bas()

Examples

```
library(MASS)
data(UScrime)

# pivot=FALSE is faster, but should only be used in full rank case
# default is pivot = TRUE
crime.bic <- bas.lm(log(y) ~ log(M) + So + log(Ed) +
  log(Po1) + log(Po2) +
  log(LF) + log(M.F) + log(Pop) + log(NW) +
  log(U1) + log(U2) + log(GDP) + log(Ineq) +
  log(Prob) + log(Time),
  data = UScrime, n.models = 2^15, prior = "BIC",
  modelprior = beta.binomial(1, 1),
  initprobs = "eplogp", pivot = FALSE
)


# use MCMC rather than enumeration
crime.mcmc <- bas.lm(log(y) ~ log(M) + So + log(Ed) +
  log(Po1) + log(Po2) +
  log(LF) + log(M.F) + log(Pop) + log(NW) +
  log(U1) + log(U2) + log(GDP) + log(Ineq) +
 log(Prob) + log(Time),
  data = UScrime,
  method = "MCMC",
  MCMC.iterations = 20000, prior = "BIC",
  modelprior = beta.binomial(1, 1),
  initprobs = "eplogp", pivot = FALSE
)

summary(crime.bic)
plot(crime.bic)
image(crime.bic, subset = -1)

# example with two-way interactions and hierarchical constraints
```

```
data(ToothGrowth)
ToothGrowth$dose <- factor(ToothGrowth$dose)
levels(ToothGrowth$dose) <- c("Low", "Medium", "High")
TG.bas <- bas.lm(len ~ supp * dose,
  data = ToothGrowth,
  modelprior = uniform(), method = "BAS",
  force.heredity = TRUE
)
summary(TG.bas)
image(TG.bas)


# don't run the following due to time limits on CRAN

## Not run:

# exmple with non-full rank case

loc <- system.file("testdata", package = "BAS")
d <- read.csv(paste(loc, "JASP-testdata.csv", sep = "/"))
fullModelFormula <- as.formula("contNormal ~  contGamma * contExpon +
                               contGamma * contcor1 + contExpon * contcor1")

# should trigger a warning (default is to use pivoting, so use pivot=FALSE
# only for full rank case)

 out = bas.lm(fullModelFormula,
              data = d,
              alpha = 0.125316,
              prior = "JZS",
              weights = facFifty, force.heredity = FALSE, pivot = FALSE)


# use pivot = TRUE to fit non-full rank case  (default)
# This is slower but safer

out =  bas.lm(fullModelFormula,
              data = d,
              alpha = 0.125316,
              prior = "JZS",
              weights = facFifty, force.heredity = FALSE, pivot = TRUE)

## End(Not run)
# more complete demo's
demo(BAS.hald)
## Not run:
demo(BAS.USCrime)

## End(Not run)
```

Bayes.outlier *Bayesian Outlier Detection*

Description

Calculate the posterior probability that the absolute value of error exceeds more than k standard deviations P(|epsilon_j| > k sigma | data) under the model Y = X B + epsilon, with epsilon ~ N(0, sigma^2 I) based on the paper by Chaloner & Brant Biometrika (1988). Either k or the prior probability of there being no outliers must be provided. This only uses the reference prior p(B, sigma) = 1; other priors and model averaging to come.

Usage

```
Bayes.outlier(lmobj, k, prior.prob)
```

Arguments

lmobj	An object of class 'lm'
k	number of standard deviations used in calculating probability of an individual case being an outlier, P(|error| > k sigma | data)
prior.prob	The prior probability of there being no outliers in the sample of size n

Value

Returns a list of three items:

e	residuals
hat	leverage values
prob.outlier	posterior probabilities of a point being an outlier
prior.prob	prior probability of a point being an outlier

References

Chaloner & Brant (1988) A Bayesian Approach to Outlier Detection and Residual Analysis Biometrika (1988) 75, 651-659

Examples

```
data("stackloss")
stack.lm <- lm(stack.loss ~ ., data = stackloss)
stack.outliers <- Bayes.outlier(stack.lm, k = 3)
plot(stack.outliers$prob.outlier, type = "h", ylab = "Posterior Probability")
# adjust for sample size for calculating prior prob that a
# a case is an outlier
stack.outliers <- Bayes.outlier(stack.lm, prior.prob = 0.95)
# cases where posterior probability exceeds prior probability
which(stack.outliers$prob.outlier > stack.outliers$prior.prob)
```

`bayesglm.fit` *Fitting Generalized Linear Models and Bayesian marginal likelihood evaluation*

Description

A version of glm.fit rewritten in C; also returns marginal likelihoods for Bayesian model comparison

Usage

```
bayesglm.fit(
  x,
  y,
  weights = rep(1, nobs),
  start = NULL,
  etastart = NULL,
  mustart = NULL,
  offset = rep(0, nobs),
  family = binomial(),
  coefprior = bic.prior(nobs),
  control = glm.control(),
  intercept = TRUE
)
```

Arguments

`x`	design matrix
`y`	response
`weights`	optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector.
`start`	starting value for coefficients in the linear predictor
`etastart`	starting values for the linear predictor
`mustart`	starting values for the vectors of means
`offset`	a priori known component to be included in the linear predictor
`family`	a description of the error distribution and link function for exponential family; currently only binomial(), poisson(), and Gamma() with canonical links are implemented.
`coefprior`	function specifying prior distribution on coefficients with optional hyperparameters leading to marginal likelihood calculations; options include `bic.prior()`, `aic.prior()`, and `ic.prior()`
`control`	a list of parameters that control convergence in the fitting process. See the documentation for `glm.control()`
`intercept`	should an intercept be included in the null model?

Details

C version of glm-fit. For different prior choices returns, marginal likelihood of model using a Laplace approximation.

Value

coefficients	MLEs
se	Standard errors of coefficients based on the sqrt of the diagonal of the inverse information matrix
mu	fitted mean
rank	numeric rank of the fitted linear model
deviance	minus twice the log likelihood evaluated at the MLEs
g	value of g in g-priors
shrinkage	shrinkage factor for coefficients in linear predictor
RegSS	quadratic form beta'I(beta)beta used in shrinkage
logmarglik	the log marginal or integrated log likelihood (up to a constant)

Author(s)

Merlise Clyde translated the glm.fit from R base into C using the .Call interface

References

glm

See Also

bic.prior

Examples

```
data(Pima.tr, package="MASS")
Y <- as.numeric(Pima.tr$type) - 1
X <- cbind(1, as.matrix(Pima.tr[,1:7]))
out <- bayesglm.fit(X, Y, family=binomial(),coefprior=bic.prior(n=length(Y)))
out$coef
out$se
# using built in function
glm(type ~ ., family=binomial(), data=Pima.tr)
```

`Bernoulli` *Independent Bernoulli Prior Distribution for Models*

Description

Creates an object representing the prior distribution on models for BAS.

Usage

```
Bernoulli(probs = 0.5)
```

Arguments

`probs`	a scalar or vector of prior inclusion probabilities. If a scalar, the values is replicated for all variables ans a 1 is added for the intercept. BAS checks to see if the length is equal to the dimension of the parameter vector for the full model and adds a 1 to include the intercept.

Details

The independent Bernoulli prior distribution is a commonly used prior in BMA, with the Uniform distribution a special case with probs=.5. If all indicator variables have a independent Bernoulli distributions with common probability probs, the distribution on model size binomial(p, probs) distribution.

Value

returns an object of class "prior", with the family and hyperparameters.

Author(s)

Merlise Clyde

See Also

`bas.lm`, `beta.binomial`,`uniform`

Other priors modelpriors: `Bernoulli.heredity()`, `beta.binomial()`, `tr.beta.binomial()`, `tr.poisson()`, `tr.power.prior()`, `uniform()`

Examples

```
Bernoulli(.9)
```

Bernoulli.heredity *Independent Bernoulli prior on models that with constraints for model hierarchy induced by interactions*

Description

Independent Bernoulli prior on models that with constraints for model hierarchy induced by interactions

Usage

```
Bernoulli.heredity(pi = 0.5, parents)
```

Arguments

pi	Bernoulli probability that term is included
parents	matrix of terms and parents with indicators of which terms are parents for each term

Note

Not implemented yet for use with bas.lm or bas.glm

See Also

Other priors modelpriors: Bernoulli(), beta.binomial(), tr.beta.binomial(), tr.poisson(), tr.power.prior(), uniform()

beta.binomial *Beta-Binomial Prior Distribution for Models*

Description

Creates an object representing the prior distribution on models for BAS.

Usage

```
beta.binomial(alpha = 1, beta = 1)
```

Arguments

alpha	parameter in the beta prior distribution
beta	parameter in the beta prior distribution

Details

The beta-binomial distribution on model size is obtained by assigning each variable inclusion indicator independent Bernoulli distributions with probability w, and then giving w a beta(alpha,beta) distribution. Marginalizing over w leads to the distribution on model size having the beta-binomial distribution. The default hyperparameters lead to a uniform distribution over model size.

Value

returns an object of class "prior", with the family and hyperparameters.

Author(s)

Merlise Clyde

See Also

bas.lm, Bernoulli,uniform

Other priors modelpriors: Bernoulli(), Bernoulli.heredity(), tr.beta.binomial(), tr.poisson(), tr.power.prior(), uniform()

Examples

```
beta.binomial(1, 10) #' @family priors modelpriors
```

`beta.prime` *Beta-Prime Prior Distribution for Coefficients in BMA Model*

Description

Creates an object representing the Beta-Prime prior that is mixture of g-priors on coefficients for BAS.

Usage

```
beta.prime(n = NULL)
```

Arguments

`n`	the sample size; if NULL, the value derived from the data in the call to ‘bas.glm‘ will be used.

Details

Creates a structure used for bas.glm.

Value

returns an object of class "prior", with the family and hyerparameters.

Author(s)

Merlise Clyde

See Also

CCH

Other beta priors: CCH(), EB.local(), IC.prior(), Jeffreys(), TG(), g.prior(), hyper.g(), hyper.g.n(), intrinsic(), robust(), tCCH(), testBF.prior()

Examples

```
beta.prime(n = 100)
```

---

bodyfat *Bodyfat Data*

Description

Lists estimates of the percentage of body fat determined by underwater weighing and various body circumference measurements for 252 men. Accurate measurement of body fat is inconvenient/costly and it is desirable to have easy methods of estimating body fat that are not inconvenient/costly.

Format

A data frame with 252 observations on the following 15 variables.

Density a numeric vector for the density determined from underwater weighing

Bodyfat percent body fat from Siri's (1956) equation

Age age of individual in years

Weight weight of the individual in pounds

Height height of individual in inches

Neck neck circumference in centimeters (cm)

Chest chest circumference (cm)

Abdomen abdomen circumference (cm)

Hip hip circumference (cm)

"Thigh" thigh circumference (cm)

"Knee" knee circumference (cm)

Ankle ankle circumference (cm)

Biceps bicep (extended) circumference (cm)

Forearm forearm circumference (cm)

Wrist wrist circumference (cm)

Details

A variety of popular health books suggest that the readers assess their health, at least in part, by estimating their percentage of body fat. In Bailey (1994), for instance, the reader can estimate body fat from tables using their age and various skin-fold measurements obtained by using a caliper. Other texts give predictive equations for body fat using body circumference measurements (e.g. abdominal circumference) and/or skin-fold measurements. See, for instance, Behnke and Wilmore (1974), pp. 66-67; Wilmore (1976), p. 247; or Katch and McArdle (1977), pp. 120-132).#

Percentage of body fat for an individual can be estimated once body density has been determined. Folks (e.g. Siri (1956)) assume that the body consists of two components - lean body tissue and fat tissue. Letting

D = Body Density (gm/cm^3) A = proportion of lean body tissue B = proportion of fat tissue (A+B=1) a = density of lean body tissue (gm/cm^3) b = density of fat tissue (gm/cm^3)

we have D = 1/[(A/a) + (B/b)] and solving for B we find B = (1/D)*[ab/(a-b)] - [b/(a-b)].

Using the estimates a=1.10 gm/cm^3 and b=0.90 gm/cm^3 (see Katch and McArdle (1977), p. 111 or Wilmore (1976), p. 123) we come up with "Siri's equation":

Percentage of Body Fat (i.e. 100*B) = 495/D - 450.#

Volume, and hence body density, can be accurately measured a variety of ways. The technique of underwater weighing "computes body volume as the difference between body weight measured in air and weight measured during water submersion. In other words, body volume is equal to the loss of weight in water with the appropriate temperature correction for the water's density" (Katch and McArdle (1977), p. 113). Using this technique,

Body Density = WA/[(WA-WW)/c.f. - LV]

where WA = Weight in air (kg) WW = Weight in water (kg) c.f. = Water correction factor (=1 at 39.2 deg F as one-gram of water occupies exactly one cm^3 at this temperature, =.997 at 76-78 deg F) LV = Residual Lung Volume (liters)

(Katch and McArdle (1977), p. 115). Other methods of determining body volume are given in Behnke and Wilmore (1974), p. 22 ff.

Measurement standards are apparently those listed in Behnke and Wilmore (1974), pp. 45-48 where, for instance, the abdomen circumference is measured "laterally, at the level of the iliac crests, and anteriorly, at the umbilicus".)

Source

These data are used to produce the predictive equations for lean body weight given in the abstract "Generalized body composition prediction equation for men using simple measurement techniques", K.W. Penrose, A.G. Nelson, A.G. Fisher, FACSM, Human Performance Research Center, Brigham Young University, Provo, Utah 84602 as listed in _Medicine and Science in Sports and Exercise_, vol. 17, no. 2, April 1985, p. 189. (The predictive equations were obtained from the first 143 of the 252 cases that are listed below). The data were generously supplied by Dr. A. Garth Fisher who gave permission to freely distribute the data and use for non-commercial purposes.

References

Bailey, Covert (1994). Smart Exercise: Burning Fat, Getting Fit, Houghton-Mifflin Co., Boston, pp. 179-186.

Behnke, A.R. and Wilmore, J.H. (1974). Evaluation and Regulation of Body Build and Composition, Prentice-Hall, Englewood Cliffs, N.J.

Siri, W.E. (1956), "Gross composition of the body", in Advances in Biological and Medical Physics, vol. IV, edited by J.H. Lawrence and C.A. Tobias, Academic Press, Inc., New York.

Katch, Frank and McArdle, William (1977). Nutrition, Weight Control, and Exercise, Houghton Mifflin Co., Boston.

Wilmore, Jack (1976). Athletic Training and Physical Fitness: Physiological Principles of the Conditioning Process, Allyn and Bacon, Inc., Boston.

Examples

```
data(bodyfat)
bodyfat.bas = bas.lm(Bodyfat ~ Abdomen, data=bodyfat, prior="ZS-null")
summary(bodyfat.bas)
plot(Bodyfat ~ Abdomen, data=bodyfat, xlab="abdomen circumference (cm)")
betas = coef(bodyfat.bas)$postmean   # current version has that intercept is ybar
betas[1] = betas[1] - betas[2]*bodyfat.bas$mean.x
abline(betas)
abline(coef(lm(Bodyfat ~ Abdomen, data=bodyfat)), col=2, lty=2)
```

CCH *Generalized g-Prior Distribution for Coefficients in BMA Models*

Description

Creates an object representing the CCH mixture of g-priors on coefficients for BAS .

Usage

```
CCH(alpha, beta, s = 0)
```

Arguments

alpha	a scalar > 0, recommended alpha=.5 (betaprime) or 1 for CCH. The hyper.g(alpha) is equivalent to CCH(alpha -2, 2, 0). Liang et al recommended values in the range 2 < alpha_h <= 4
beta	a scalar > 0. The value is not updated by the data; beta should be a function of n for consistency under the null model. The hyper-g corresponds to b = 2
s	a scalar, recommended s=0

Details

Creates a structure used for bas.glm.

Value

returns an object of class "prior", with the family and hyperparameters.

Author(s)

Merlise A Clyde

See Also

IC.prior, bic.prior, bas.glm

Other beta priors: EB.local(), IC.prior(), Jeffreys(), TG(), beta.prime(), g.prior(), hyper.g(), hyper.g.n(), intrinsic(), robust(), tCCH(), testBF.prior()

Examples

```
CCH(alpha = .5, beta = 100, s = 0)
```

---

climate *Climate Data*

Description

Climate Data

Format

Scientists are interested in the Earth's temperature change since the last glacial maximum, about 20,000 years ago. The first study to estimate the temperature change was published in 1980, and estimated a change of -1.5 degrees C, +/- 1.2 degrees C in tropical sea surface temperatures. The negative value means that the Earth was colder then than now. Since 1980 there have been many other studies. `climate` is a dataset with 63 measurements on 5 variables:

deltaT the response variables, which is the change in temperature in degrees Celsius;

sdev a standard deviation for the calculated *deltaT*;

proxy a number 1-8 reflecting which type of measurement system was used to derive deltaT. Some proxies can be used over land, others over water. The proxies are coded as
1 "Mg/Ca"
2 "alkenone"
3 "Faunal"
4 "Sr/Ca"
5 "del 180"
6 "Ice Core"
7 "Pollen"
8 "Noble Gas"

T/M , an indicator of whether it was a terrestrial or marine study (T/M), which is coded as 0 for Terrestrial, 1 for Marine;

latitude the latitude where the data were collected.

Source

Data provided originally by Michael Lavine

coef.bas *Coefficients of a Bayesian Model Average object*

Description

Extract conditional posterior means and standard deviations, marginal posterior means and standard deviations, posterior probabilities, and marginal inclusions probabilities under Bayesian Model Averaging from an object of class 'bas'

Usage

```
## S3 method for class 'bas'
coef(object, n.models, estimator = "BMA", ...)

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

Arguments

object	object of class 'bas' created by BAS
n.models	Number of top models to report in the printed summary, for coef the default is to use all models. To extract summaries for the Highest Probability Model, use n.models=1 or estimator="HPM".
estimator	return summaries for a selected model, rather than using BMA. Options are 'HPM' (highest posterior probability model) ,'MPM' (median probability model), and 'BMA'
...	other optional arguments
x	object of class 'coef.bas' to print
digits	number of significant digits to print

Details

Calculates posterior means and (approximate) standard deviations of the regression coefficients under Bayesian Model averaging using g-priors and mixtures of g-priors. Print returns overall summaries. For fully Bayesian methods that place a prior on g, the posterior standard deviations do not take into account full uncertainty regarding g. Will be updated in future releases.

Value

coefficients returns an object of class coef.bas with the following:

conditionalmeans	a matrix with conditional posterior means for each model
conditionalsd	standard deviations for each model
postmean	marginal posterior means of each regression coefficient using BMA

postsd	marginal posterior standard deviations using BMA
postne0	vector of posterior inclusion probabilities, marginal probability that a coefficient is non-zero

Note

With highly correlated variables, marginal summaries may not be representative of the joint distribution. Use plot.coef.bas to view distributions. The value reported for the intercept is under the centered parameterization. Under the Gaussian error model it will be centered at the sample mean of Y.

Author(s)

Merlise Clyde <clyde@duke.edu>

References

Liang, F., Paulo, R., Molina, G., Clyde, M. and Berger, J.O. (2005) Mixtures of g-priors for Bayesian Variable Selection. Journal of the American Statistical Association. 103:410-423. doi:10.1198/016214507000001337

See Also

bas, confint.coef.bas

Other bas methods: BAS, bas.lm(), confint.coef.bas(), confint.pred.bas(), diagnostics(), fitted.bas(), force.heredity.bas(), image.bas(), plot.confint.bas(), predict.bas(), predict.basglm(), summary.bas(), update.bas(), variable.names.pred.bas()

Examples

```
data("Hald")
hald.gprior =  bas.lm(Y~ ., data=Hald, n.models=2^4, alpha=13,
                      prior="ZS-null", initprobs="Uniform", update=10)
coef.hald.gprior = coefficients(hald.gprior)
coef.hald.gprior
plot(coef.hald.gprior)
confint(coef.hald.gprior)

#Estimation under Median Probability Model
coef.hald.gprior = coefficients(hald.gprior, estimator="MPM")
coef.hald.gprior
plot(coef.hald.gprior)
plot(confint(coef.hald.gprior))


coef.hald.gprior = coefficients(hald.gprior, estimator="HPM")
coef.hald.gprior
plot(coef.hald.gprior)
confint(coef.hald.gprior)

# To add estimation under Best Predictive Model
```

`confint.coef.bas` *Compute Credible Intervals for BAS regression coefficients from BAS objects*

Description

Uses Monte Carlo simulations using posterior means and standard deviations of coefficients to generate draws from the posterior distributions and returns highest posterior density (HPD) credible intervals. If the number of models equals one, then use the t distribution to find intervals. These currently condition on the estimate of g. than the description above ~~

Usage

```
## S3 method for class 'coef.bas'
confint(object, parm, level = 0.95, nsim = 10000, ...)
```

Arguments

`object`	a coef.bas object
`parm`	a specification of which parameters are to be given credible intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
`level`	the probability coverage required
`nsim`	number of Monte Carlo draws from the posterior distribution. Used when number of models is greater than 1.
`...`	other arguments to passed; none currently

Value

A matrix (or vector) with columns giving lower and upper HPD credible limits for each parameter. These will be labeled as 1-level)/2 and 1 - (1-level)/2 in percent (by default 2.5 and 97.5).

Note

For mixture of g-priors these are approximate. This uses Monte Carlo sampling so results may be subject to Monte Carlo variation and larger values of nsim may be needed to reduce variability.

Author(s)

Merlise A Clyde

See Also

Other CI methods: confint.pred.bas(), plot.confint.bas()

Other bas methods: BAS, bas.lm(), coef.bas(), confint.pred.bas(), diagnostics(), fitted.bas(), force.heredity.bas(), image.bas(), plot.confint.bas(), predict.bas(), predict.basglm(), summary.bas(), update.bas(), variable.names.pred.bas()

Examples

```
data("Hald")
hald_gprior <-  bas.lm(Y~ ., data=Hald, alpha=13,
                           prior="g-prior")
coef_hald <- coef(hald_gprior)
confint(coef_hald)
confint(coef_hald, approx=FALSE, nsim=5000)
# extract just the coefficient of X4
confint(coef_hald, parm="X4")
```

---

confint.pred.bas *Compute Credible (Bayesian Confidence) Intervals for a BAS predict object*

Description

Compute credible intervals for in-sample or out of sample prediction or for the regression function

Usage

```
## S3 method for class 'pred.bas'
confint(object, parm, level = 0.95, nsim = 10000, ...)
```

Arguments

object	an object created by predict.bas
parm	character variable, "mean" or "pred". If missing parm='pred'.
level	the nominal level of the (point-wise) credible interval
nsim	number of Monte Carlo simulations for sampling methods with BMA
...	optional arguments to pass on to next function call; none at this time.

Details

This constructs approximate 95 percent Highest Posterior Density intervals for 'pred.bas' objects. If the estimator is based on model selection, the intervals use a Student t distribution using the estimate of g. If the estimator is based on BMA, then nsim draws from the mixture of Student t distributions are obtained with the HPD interval obtained from the Monte Carlo draws.

Value

a matrix with lower and upper level * 100 percent credible intervals for either the mean of the regression function or predicted values.

Author(s)

Merlise A Clyde

See Also

predict.bas

Other bas methods: BAS, bas.lm(), coef.bas(), confint.coef.bas(), diagnostics(), fitted.bas(), force.heredity.bas(), image.bas(), plot.confint.bas(), predict.bas(), predict.basglm(), summary.bas(), update.bas(), variable.names.pred.bas()

Other CI methods: confint.coef.bas(), plot.confint.bas()

Examples

```
data("Hald")
hald.gprior =  bas.lm(Y~ ., data=Hald, alpha=13, prior="g-prior")
hald.pred = predict(hald.gprior, estimator="BPM", predict=FALSE, se.fit=TRUE)
confint(hald.pred, parm="mean")
confint(hald.pred)  #default
hald.pred = predict(hald.gprior, estimator="BMA", predict=FALSE, se.fit=TRUE)
confint(hald.pred)
```

---

cv.summary.bas *Summaries for Out of Sample Prediction*

Description

Compute average prediction error from out of sample predictions

Usage

```
cv.summary.bas(pred, ytrue, score = "squared-error")
```

Arguments

pred	fitted or predicted value from the output from predict.bas
ytrue	vector of left out response values
score	function used to summarize error rate. Either "squared-error", or "miss-class"

Value

For squared error, the average prediction error for the Bayesian estimator error = sqrt(sum(ytrue - yhat)^2/npred) while for binary data the misclassification rate is more appropriate.

Author(s)

Merlise Clyde <clyde@duke.edu>

See Also

predict.bas

Examples

```
## Not run:
library(foreign)
cognitive <- read.dta("https://www.stat.columbia.edu/~gelman/arm/examples/child.iq/kidiq.dta")
cognitive$mom_work <- as.numeric(cognitive$mom_work > 1)
cognitive$mom_hs <- as.numeric(cognitive$mom_hs > 0)
colnames(cognitive) <- c("kid_score", "hs", "iq", "work", "age")

set.seed(42)
n <- nrow(cognitive)
test <- sample(1:n, size = round(.20 * n), replace = FALSE)
testdata <- cognitive[test, ]
traindata <- cognitive[-test, ]
cog_train <- bas.lm(kid_score ~ ., prior = "BIC", modelprior = uniform(), data = traindata)
yhat <- predict(cog_train, newdata = testdata, estimator = "BMA", se = F)
cv.summary.bas(yhat$fit, testdata$kid_score)

## End(Not run)
```

---

diagnostics *BAS MCMC diagnostic plot*

Description

Function to help assess convergence of MCMC sampling for bas objects.

Usage

```
diagnostics(obj, type = c("pip", "model"), ...)
```

Arguments

obj	an object created by bas.lm or bas.glm
type	type of diagnostic plot. If "pip" the marginal inclusion probabilities are used, while if "model", plot posterior model probabilities
...	additional graphics parameters to be passed to plot

Details

BAS calculates posterior model probabilities in two ways when method="MCMC". The first is using the relative Monte Carlo frequencies of sampled models. The second is to renormalize the marginal likelihood times prior probabilities over the sampled models. If the Markov chain has converged, these two quantities should be the same and fall on a 1-1 line. If not, running longer may be required. If the chain has not converged, the Monte Carlo frequencies may have less bias, although may exhibit more variability on repeated runs.

Value

a plot with of the marginal inclusion probabilities (pip) estimated by MCMC and renormalized marginal likelihoods times prior probabilities or model probabilities.

Author(s)

Merlise Clyde (<clyde@duke.edu>)

See Also

Other bas methods: BAS, bas.lm(), coef.bas(), confint.coef.bas(), confint.pred.bas(), fitted.bas(), force.heredity.bas(), image.bas(), plot.confint.bas(), predict.bas(), predict.basglm(), summary.bas(), update.bas(), variable.names.pred.bas()

Examples

```
library(MASS)
data(UScrime)
UScrime[, -2] <- log(UScrime[, -2])
crime.ZS <- bas.lm(y ~ .,
  data = UScrime,
  prior = "ZS-null",
  modelprior = uniform(),
  method = "MCMC",
  MCMC.iter = 1000
) # short run for the example
diagnostics(crime.ZS)
```

EB.global *Find the global Empirical Bayes estimates for BMA*

Description

Finds the global Empirical Bayes estimates of g in Zellner's g-prior and model probabilities

Usage

```
EB.global(object, tol = 0.1, g.0 = NULL, max.iterations = 100)
```

Arguments

object	A 'bas' object created by bas
tol	tolerance for estimating g
g.0	initial value for g
max.iterations	Maximum number of iterations for the EM algorithm

Details

Uses the EM algorithm in Liang et al to estimate the type II MLE of g in Zellner's g prior

Value

An object of class 'bas' using Zellner's g prior with an estimate of g based on all models

Author(s)

Merlise Clyde <clyde@stat.duke.edu>

References

Liang, F., Paulo, R., Molina, G., Clyde, M. and Berger, J.O. (2008) Mixtures of g-priors for Bayesian Variable Selection. Journal of the American Statistical Association. 103:410-423. doi:10.1198/016214507000001337

See Also

bas, update

Examples

```
library(MASS)
data(UScrime)
UScrime[,-2] = log(UScrime[,-2])
# EB local uses a different g within each model
crime.EBL =  bas.lm(y ~ ., data=UScrime, n.models=2^15,
                    prior="EB-local", initprobs= "eplogp")
# use a common (global) estimate of g
crime.EBG = EB.global(crime.EBL)
```

EB.local — *Empirical Bayes Prior Distribution for Coefficients in BMA Model*

Description

Creates an object representing the EB prior for BAS GLM.

Usage

```
EB.local()
```

Details

Creates a structure used for bas.glm.

Value

returns an object of class "prior", with the family and hyerparameters.

Author(s)

Merlise Clyde

See Also

CCH and bas.glm

Other beta priors: CCH(), IC.prior(), Jeffreys(), TG(), beta.prime(), g.prior(), hyper.g(), hyper.g.n(), intrinsic(), robust(), tCCH(), testBF.prior()

Examples

```
EB.local()
```

eplogprob — *eplogprob - Compute approximate marginal inclusion probabilities from pvalues*

Description

eplogprob calculates approximate marginal posterior inclusion probabilities from p-values computed from a linear model using a lower bound approximation to Bayes factors. Used to obtain initial inclusion probabilities for sampling using Bayesian Adaptive Sampling bas.lm

Usage

```
eplogprob(lm.obj, thresh = 0.5, max = 0.99, int = TRUE)
```

Arguments

lm.obj	a linear model object
thresh	the value of the inclusion probability when if the p-value > 1/exp(1), where the lower bound approximation is not valid.
max	maximum value of the inclusion probability; used for the bas.lm function to keep initial inclusion probabilities away from 1.
int	If the Intercept is included in the linear model, set the marginal inclusion probability corresponding to the intercept to 1

Details

Sellke, Bayarri and Berger (2001) provide a simple calibration of p-values

BF(p) = -e p log(p)

which provide a lower bound to a Bayes factor for comparing H0: beta = 0 versus H1: beta not equal to 0, when the p-value p is less than 1/e. Using equal prior odds on the hypotheses H0 and H1, the approximate marginal posterior inclusion probability

p(beta != 0 | data) = 1/(1 + BF(p))

When p > 1/e, we set the marginal inclusion probability to 0.5 or the value given by thresh.

Value

eplogprob returns a vector of marginal posterior inclusion probabilities for each of the variables in the linear model. If int = TRUE, then the inclusion probability for the intercept is set to 1. If the model is not full rank, variables that are linearly dependent base on the QR factorization will have NA for their p-values. In bas.lm, where the probabilities are used for sampling, the inclusion probability is set to 0.

Author(s)

Merlise Clyde <clyde@stat.duke.edu>

References

Sellke, Thomas, Bayarri, M. J., and Berger, James O. (2001), "Calibration of p-values for testing precise null hypotheses", The American Statistician, 55, 62-71.

See Also

bas

Examples

```
library(MASS)
data(UScrime)
UScrime[,-2] = log(UScrime[,-2])
eplogprob(lm(y ~ ., data=UScrime))
```

`eplogprob.marg` *eplogprob.marg - Compute approximate marginal inclusion probabilities from pvalues*

Description

`eplogprob.marg` calculates approximate marginal posterior inclusion probabilities from p-values computed from a series of simple linear regression models using a lower bound approximation to Bayes factors. Used to order variables and if appropriate obtain initial inclusion probabilities for sampling using Bayesian Adaptive Sampling `bas.lm`

Usage

```
eplogprob.marg(Y, X, thresh = 0.5, max = 0.99, int = TRUE)
```

Arguments

`Y`	response variable
`X`	design matrix with a column of ones for the intercept
`thresh`	the value of the inclusion probability when if the p-value > 1/exp(1), where the lower bound approximation is not valid.
`max`	maximum value of the inclusion probability; used for the `bas.lm` function to keep initial inclusion probabilities away from 1.
`int`	If the Intercept is included in the linear model, set the marginal inclusion probability corresponding to the intercept to 1

Details

Sellke, Bayarri and Berger (2001) provide a simple calibration of p-values

BF(p) = -e p log(p)

which provide a lower bound to a Bayes factor for comparing H0: beta = 0 versus H1: beta not equal to 0, when the p-value p is less than 1/e. Using equal prior odds on the hypotheses H0 and H1, the approximate marginal posterior inclusion probability

p(beta != 0 | data) = 1/(1 + BF(p))

When p > 1/e, we set the marginal inclusion probability to 0.5 or the value given by `thresh`. For the eplogprob.marg the marginal p-values are obtained using statistics from the p simple linear regressions

P(F > (n-2) R2/(1 - R2)) where F ~ F(1, n-2) where R2 is the square of the correlation coefficient between y and X_j.

Value

`eplogprob.prob` returns a vector of marginal posterior inclusion probabilities for each of the variables in the linear model. If int = TRUE, then the inclusion probability for the intercept is set to 1.

Author(s)

Merlise Clyde <clyde@stat.duke.edu>

References

Sellke, Thomas, Bayarri, M. J., and Berger, James O. (2001), "Calibration of p-values for testing precise null hypotheses", The American Statistician, 55, 62-71.

See Also

bas

Examples

```
library(MASS)
data(UScrime)
UScrime[,-2] = log(UScrime[,-2])
eplogprob(lm(y ~ ., data=UScrime))
```

---

fitted.bas *Fitted values for a BAS BMA objects*

Description

Calculate fitted values for a BAS BMA object

Usage

```
## S3 method for class 'bas'
fitted(
  object,
  type = "link",
  estimator = "BMA",
  top = NULL,
  na.action = na.pass,
  ...
)
```

Arguments

object	An object of class 'bas' as created by bas
type	type equals "response" or "link" in the case of GLMs (default is 'link')
estimator	estimator type of fitted value to return. Default is to use BMA with all models. Options include 'HPM' the highest probability model 'BMA' Bayesian model averaging, using optionally only the 'top' models 'MPM' the median probability model of Barbieri and Berger. 'BPM' the model that is closest to BMA predictions under squared error loss

top	optional argument specifying that the 'top' models will be used in constructing the BMA prediction, if NULL all models will be used. If top=1, then this is equivalent to 'HPM'
na.action	function determining what should be done with missing values in newdata. The default is to predict NA.
...	optional arguments, not used currently

Details

Calculates fitted values at observed design matrix using either the highest probability model, 'HPM', the posterior mean (under BMA) 'BMA', the median probability model 'MPM' or the best predictive model 'BPM". The median probability model is defined by including variable where the marginal inclusion probability is greater than or equal to 1/2. For type="BMA", the weighted average may be based on using a subset of the highest probability models if an optional argument is given for top. By default BMA uses all sampled models, which may take a while to compute if the number of variables or number of models is large. The "BPM" is found be computing the squared distance of the vector of fitted values for a model and the fitted values under BMA and returns the model with the smallest distance. In the presence of multicollinearity this may be quite different from the MPM, with extreme collinearity may drop relevant predictors.

Value

A vector of length n of fitted values.

Author(s)

Merlise Clyde <clyde@duke.edu>

References

Barbieri, M. and Berger, J.O. (2004) Optimal predictive model selection. Annals of Statistics. 32, 870-897.
https://projecteuclid.org/euclid.aos/1085408489&url=/UI/1.0/Summarize/euclid.aos/1085408489

Clyde, M. Ghosh, J. and Littman, M. (2010) Bayesian Adaptive Sampling for Variable Selection and Model Averaging. Journal of Computational Graphics and Statistics. 20:80-101
doi:10.1198/jcgs.2010.09049

See Also

predict.bas predict.basglm

Other bas methods: BAS, bas.lm(), coef.bas(), confint.coef.bas(), confint.pred.bas(), diagnostics(), force.heredity.bas(), image.bas(), plot.confint.bas(), predict.bas(), predict.basglm(), summary.bas(), update.bas(), variable.names.pred.bas()

Other predict methods: predict.bas(), predict.basglm(), variable.names.pred.bas()

Examples

```
data(Hald)
hald.gprior =  bas.lm(Y~ ., data=Hald, prior="ZS-null", initprobs="Uniform")
plot(Hald$Y, fitted(hald.gprior, estimator="HPM"))
plot(Hald$Y, fitted(hald.gprior, estimator="BMA", top=3))
plot(Hald$Y, fitted(hald.gprior, estimator="MPM"))
plot(Hald$Y, fitted(hald.gprior, estimator="BPM"))
```

`force.heredity.bas` *Post processing function to force constraints on interaction inclusion bas BMA objects*

Description

This function takes the output of a bas object and allows higher order interactions to be included only if their parent lower order interactions terms are in the model, by assigning zero prior probability, and hence posterior probability, to models that do not include their respective parents.

Usage

```
force.heredity.bas(object, prior.prob = 0.5)
```

Arguments

`object`	a bas linear model or generalized linear model object
`prior.prob`	prior probability that a term is included conditional on parents being included

Value

a bas object with updated models, coefficients and summaries obtained removing all models with zero prior and posterior probabilities.

Note

Currently prior probabilities are computed using conditional Bernoulli distributions, i.e. P(gamma_j = 1 | Parents(gamma_j) = 1) = prior.prob. This is not very efficient for models with a large number of levels. Future updates will force this at the time of sampling.

Author(s)

Merlise A Clyde

See Also

Other bas methods: `BAS`, `bas.lm()`, `coef.bas()`, `confint.coef.bas()`, `confint.pred.bas()`, `diagnostics()`, `fitted.bas()`, `image.bas()`, `plot.confint.bas()`, `predict.bas()`, `predict.basglm()`, `summary.bas()`, `update.bas()`, `variable.names.pred.bas()`

Examples

```
data("chickwts")
bas.chk <- bas.lm(weight ~ feed, data = chickwts)
#  summary(bas.chk)  # 2^5 = 32 models
bas.chk.int <- force.heredity.bas(bas.chk)
#  summary(bas.chk.int)  # two models now


data(Hald)
bas.hald <- bas.lm(Y ~ .^2, data = Hald)
bas.hald.int <- force.heredity.bas(bas.hald)
image(bas.hald.int)

image(bas.hald.int)

# two-way interactions
data(ToothGrowth)
ToothGrowth$dose <- factor(ToothGrowth$dose)
levels(ToothGrowth$dose) <- c("Low", "Medium", "High")
TG.bas <- bas.lm(len ~ supp * dose, data = ToothGrowth, modelprior = uniform())
TG.bas.int <- force.heredity.bas(TG.bas)
image(TG.bas.int)
```

---

g.prior *Families of G-Prior Distribution for Coefficients in BMA Models*

Description

Creates an object representing the g-prior distribution on coefficients for BAS.

Usage

```
g.prior(g)
```

Arguments

g	a scalar used in the covariance of Zellner's g-prior, Cov(beta) = sigma^2 g (X'X)^-1

Details

Creates a structure used for BAS.

Value

returns an object of class "prior", with the family and hyerparameters.

Author(s)

Merlise Clyde

See Also

IC.prior

Other beta priors: CCH(), EB.local(), IC.prior(), Jeffreys(), TG(), beta.prime(), hyper.g(), hyper.g.n(), intrinsic(), robust(), tCCH(), testBF.prior()

Examples

```
g.prior(100)
```

Hald *Hald Data*

Description

The Hald data have been used in many books and papers to illustrate variable selection. The data relate to an engineering application that was concerned with the effect of the composition of cement on heat evolved during hardening. The response variable *Y* is the *heat evolved* in a cement mix. The four explanatory variables are ingredients of the mix, X1: *tricalcium aluminate*, X2: *tricalcium silicate*, X3: *tetracalcium alumino ferrite*, X4: *dicalcium silicate*. An important feature of these data is that the variables X1 and X3 are highly correlated, as well as the variables X2 and X4. Thus we should expect any subset of (X1,X2,X3,X4) that includes one variable from highly correlated pair to do as any subset that also includes the other member.

Format

hald is a dataframe with 13 observations and 5 variables (columns),

Y: Heat evolved per gram of cement (in calories) X1: Amount of tricalcium aluminate X2: Amount of tricalcium silicate X3: Amount of tetracalcium alumino ferrite X4: Amount of dicalcium silicate

Source

Wood, H., Steinour, H.H., and Starke, H.R. (1932). "Effect of Composition of Portland cement on Heat Evolved During Hardening", Industrial and Engineering Chemistry, 24, 1207-1214.

---

hyper.g *Hyper-g-Prior Distribution for Coefficients in BMA Models*

Description

Creates an object representing the hyper-g mixture of g-priors on coefficients for BAS.

Usage

```
hyper.g(alpha = 3)
```

Arguments

alpha	a scalar > 0. The hyper.g(alpha) is equivalent to CCH(alpha -2, 2, 0). Liang et al recommended values in the range 2 < alpha_h <= 3

Details

Creates a structure used for bas.glm.

Value

returns an object of class "prior", with the family and hyerparameters.

Author(s)

Merlise Clyde

See Also

CCH bas.glm

Other beta priors: CCH(), EB.local(), IC.prior(), Jeffreys(), TG(), beta.prime(), g.prior(), hyper.g.n(), intrinsic(), robust(), tCCH(), testBF.prior()

Examples

```
hyper.g(alpha = 3)
```

---

hyper.g.n *Generalized hyper-g/n Prior Distribution for g for mixtures of g-priors on Coefficients in BMA Models*

Description

Creates an object representing the hyper-g/n mixture of g-priors on coefficients for BAS. This is a special case of the tCCH prior

Usage

```
hyper.g.n(alpha = 3, n = NULL)
```

Arguments

alpha	a scalar > 0, recommended 2 < alpha <= 3
n	The sample size; if NULL, the value derived from the data in the call to ‘bas.glm‘ will be used.

Details

Creates a structure used for bas.glm. This is a special case of the tCCH, where hyper.g.n(alpha=3, n) is equivalent to tCCH(alpha=1, beta=2, s=0, r=1.5, v = 1, theta=1/n)

Value

returns an object of class "prior", with the family and hyerparameters.

Author(s)

Merlise Clyde

See Also

tCCH, robust, hyper.g, CCHbas.glm

Other beta priors: CCH(), EB.local(), IC.prior(), Jeffreys(), TG(), beta.prime(), g.prior(), hyper.g(), intrinsic(), robust(), tCCH(), testBF.prior()

Examples

```
n <- 500
hyper.g.n(alpha = 3, n = n)
```

---

hypergeometric1F1 *Confluent hypergeometric1F1 function*

Description

Compute the Confluent Hypergeometric function: 1F1(a,b,c,t) = Gamma(b)/(Gamma(b-a)Gamma(a)) Int_0^1 t^(a-1) (1 - t)^(b-a-1) exp(c t) dt

Usage

```
hypergeometric1F1(a, b, c, laplace = FALSE, log = TRUE)
```

Arguments

a	arbitrary
b	Must be greater 0
c	arbitrary
laplace	The default is to use the Cephes library; for large a or s this may return an NA, Inf or negative values,, in which case you should use the Laplace approximation.
log	if TRUE, return log(1F1)

Author(s)

Merlise Clyde (<clyde@stat.duke.edu>)

References

Cephes library hyp1f1.c

See Also

Other special functions: `hypergeometric2F1()`, `phi1()`, `trCCH()`

Examples

```
hypergeometric1F1(11.14756, 0.5, 0.00175097)
```

`hypergeometric2F1` *Gaussian hypergeometric2F1 function*

Description

Compute the Gaussian Hypergeometric2F1 function: 2F1(a,b,c,z) = Gamma(b-c) Int_0^1 t^(b-1) (1 - t)^(c -b -1) (1 - t z)^(-a) dt

Usage

```
hypergeometric2F1(a, b, c, z, method = "Cephes", log = TRUE)
```

Arguments

a	arbitrary
b	Must be greater 0
c	Must be greater than b if |z| < 1, and c > b + a if z = 1
z	|z| <= 1
method	The default is to use the Cephes library routine. This sometimes is unstable for large a or z near one returning Inf or negative values. In this case, try method="Laplace", which use a Laplace approximation for tau = exp(t/(1-t)).
log	if TRUE, return log(2F1)

Details

The default is to use the routine hyp2f1.c from the Cephes library. If that return a negative value or Inf, one should try method="Laplace" which is based on the Laplace approximation as described in Liang et al JASA 2008. This is used in the hyper-g prior to calculate marginal likelihoods.

Value

if log=T returns the log of the 2F1 function; otherwise the 2F1 function.

Author(s)

Merlise Clyde (<clyde@duke.edu>)

References

Cephes library hyp2f1.c

Liang, F., Paulo, R., Molina, G., Clyde, M. and Berger, J.O. (2005) Mixtures of g-priors for Bayesian Variable Selection. Journal of the American Statistical Association. 103:410-423. doi:10.1198/016214507000001337

See Also

Other special functions: `hypergeometric1F1()`, `phi1()`, `trCCH()`

Examples

```
hypergeometric2F1(12, 1, 2, .65)
```

IC.prior *Information Criterion Families of Prior Distribution for Coefficients in BMA Models*

Description

Creates an object representing the prior distribution on coefficients for BAS.

Usage

```
IC.prior(penalty)
```

Arguments

`penalty`	a scalar used in the penalized loglikelihood of the form penalty*dimension

Details

The log marginal likelihood is approximated as -2*(deviance + penalty*dimension). Allows alternatives to AIC (penalty = 2) and BIC (penalty = log(n)). For BIC, the argument may be missing, in which case the sample size is determined from the call to 'bas.glm' and used to determine the penalty.

Value

returns an object of class "prior", with the family and hyerparameters.

Author(s)

Merlise Clyde

See Also

g.prior

Other beta priors: CCH(), EB.local(), Jeffreys(), TG(), beta.prime(), g.prior(), hyper.g(), hyper.g.n(), intrinsic(), robust(), tCCH(), testBF.prior()

Examples

```
IC.prior(2)
aic.prior()
bic.prior(100)
```

---

image.bas *Images of models used in Bayesian model averaging*

Description

Creates an image of the models selected using bas.

Usage

```
## S3 method for class 'bas'
image(
  x,
  top.models = 20,
  intensity = TRUE,
  prob = TRUE,
  log = TRUE,
  rotate = TRUE,
  color = "rainbow",
  subset = NULL,
  drop.always.included = FALSE,
  offset = 0.75,
  digits = 3,
  vlas = 2,
  plas = 0,
  rlas = 0,
  ...
)
```

Arguments

x	A BMA object of type 'bas' created by BAS
top.models	Number of the top ranked models to plot
intensity	Logical variable, when TRUE image intensity is proportional to the probability or log(probability) of the model, when FALSE, intensity is binary indicating just presence (light) or absence (dark) of a variable.

prob	Logical variable for whether the area in the image for each model should be proportional to the posterior probability (or log probability) of the model (TRUE) or with equal area (FALSE).
log	Logical variable indicating whether the intensities should be based on log posterior odds (TRUE) or posterior probabilities (FALSE). The log of the posterior odds is for comparing the each model to the worst model in the top.models.
rotate	Should the image of models be rotated so that models are on the y-axis and variables are on the x-axis (TRUE)
color	The color scheme for image intensities. The value "rainbow" uses the rainbow palette. The value "blackandwhite" produces a black and white image (greyscale image)
subset	indices of variables to include/exclude in plot
drop.always.included	logical variable to drop variables that are always forced into the model. FALSE by default.
offset	numeric value to add to intensity
digits	number of digits in posterior probabilities to keep
vlas	las parameter for placing variable names; see par
plas	las parameter for posterior probability axis
rlas	las parameter for model ranks
...	Other parameters to be passed to the image and axis functions.

Details

Creates an image of the model space sampled using bas. If a subset of the top models are plotted, then probabilities are renormalized over the subset.

Note

Suggestion to allow area of models be proportional to posterior probability due to Thomas Lumley

Author(s)

Merlise Clyde <clyde@stat.duke.edu>

References

Clyde, M. (1999) Bayesian Model Averaging and Model Search Strategies (with discussion). In Bayesian Statistics 6. J.M. Bernardo, A.P. Dawid, J.O. Berger, and A.F.M. Smith eds. Oxford University Press, pages 157-185.

See Also

bas

Other bas methods: BAS, bas.lm(), coef.bas(), confint.coef.bas(), confint.pred.bas(), diagnostics(), fitted.bas(), force.heredity.bas(), plot.confint.bas(), predict.bas(), predict.basglm(), summary.bas(), update.bas(), variable.names.pred.bas()

Other bas plots: plot.bas(), plot.coef.bas()

Examples

```
require(graphics)
data("Hald")
hald.ZSprior <- bas.lm(Y ~ ., data = Hald, prior = "ZS-null")
image(hald.ZSprior, drop.always.included = TRUE) # drop the intercept
```

`intrinsic` *Intrinsic Prior Distribution for Coefficients in BMA Models*

Description

Creates an object representing the intrinsic prior on g, a special case of the tCCH mixture of g-priors on coefficients for BAS.

Usage

```
intrinsic(n = NULL)
```

Arguments

`n`	the sample size; if NULL, the value derived from the data in the call to 'bas.glm' will be used.

Details

Creates a structure used for `bas.glm`.

Value

returns an object of class "prior", with the family "intrinsic" of class "TCCH" and hyperparameters alpha = 1, beta = 1, s = 0, r = 1, n = n for the tCCH prior where theta in the tCCH prior is determined by the model size and sample size.

Author(s)

Merlise A Clyde

References

Womack, A., Novelo,L.L., Casella, G. (2014). "Inference From Intrinsic Bayes' Procedures Under Model Selection and Uncertainty". Journal of the American Statistical Association. 109:1040-1053. doi:10.1080/01621459.2014.880348

See Also

`tCCH`, `robust`, `hyper.g`, `hyper.g.nbas.glm`

Other beta priors: `CCH()`, `EB.local()`, `IC.prior()`, `Jeffreys()`, `TG()`, `beta.prime()`, `g.prior()`, `hyper.g()`, `hyper.g.n()`, `robust()`, `tCCH()`, `testBF.prior()`

Examples

```
n <- 500
tCCH(alpha = 1, beta = 2, s = 0, r = 1.5, v = 1, theta = 1 / n)
```

`Jeffreys` *Jeffreys Prior Distribution for g for Mixtures of g-Priors for Coefficients in BMA Models*

Description

Creates an object representing the Jeffrey's Prior on g mixture of g-priors on coefficients for BAS. This is equivalent to a limiting version of the CCH(a, 2, 0) with a = 0 or they hyper-g(a = 2) and is an improper prior. As g does not appear in the Null Model, Bayes Factors and model probabilities are not well-defined because of arbitrary normalizing constants, and for this reason the null model is excluded and the same constants are used across other models.

Usage

```
Jeffreys()
```

Details

Creates a structure used for `bas.glm`.

Value

returns an object of class "prior", with the family and hyerparameters.

Author(s)

Merlise Clyde

See Also

`CCH` `bas.glm`

Other beta priors: `CCH()`, `EB.local()`, `IC.prior()`, `TG()`, `beta.prime()`, `g.prior()`, `hyper.g()`, `hyper.g.n()`, `intrinsic()`, `robust()`, `tCCH()`, `testBF.prior()`

Examples

```
Jeffreys()
```

`list2matrix.bas` *Coerce a BAS list object into a matrix.*

Description

Models, coefficients, and standard errors in objects of class 'bas' are represented as a list of lists to reduce storage by omitting the zero entries. These functions coerce the list object to a matrix and fill in the zeros to facilitate other computations.

Usage

```
list2matrix.bas(x, what, which.models = NULL)
```

Arguments

`x`	a 'bas' object
`what`	name of bas list to coerce
`which.models`	a vector of indices use to extract a subset

Details

`list2matrix.bas(x, which)` is equivalent to `list2matrix.which(x)`, however, the latter uses sapply rather than a loop. `list2matrix.which` and `which.matrix` both coerce `x$which` into a matrix.

Value

a matrix representation of `x$what`, with number of rows equal to the length of which.models or total number of models and number of columns `x$n.vars`

Author(s)

Merlise Clyde <clyde@duke.edu>

See Also

`bas`

Other as.matrix methods: `list2matrix.which()`, `which.matrix()`

Examples

```
data(Hald)
hald.bic <-  bas.lm(Y ~ ., data=Hald, prior="BIC",
                    initprobs= "eplogp")
coef <- list2matrix.bas(hald.bic, "mle")  # extract all coefficients
se <- list2matrix.bas(hald.bic, "mle.se")
models <- list2matrix.which(hald.bic)     #matrix of model indicators
models <- which.matrix(hald.bic$which, hald.bic$n.vars)     #matrix of model indicators
```

`list2matrix.which` *Coerce a BAS list object into a matrix.*

Description

Models, coefficients, and standard errors in objects of class 'bas' are represented as a list of lists to reduce storage by omitting the zero entries. These functions coerce the list object to a matrix and fill in the zeros to facilitate other computations.

Usage

```
list2matrix.which(x, which.models = NULL)
```

Arguments

`x`	a 'bas' object
`which.models`	a vector of indices use to extract a subset

Details

`list2matrix.bas(x, which)` is equivalent to `list2matrix.which(x)`, however, the latter uses sapply rather than a loop. `list2matrix.which` and `which.matrix` both coerce `x$which` into a matrix.

Value

a matrix representation of `x$what`, with number of rows equal to the length of which.models or total number of models and number of columns `x$n.vars`

Author(s)

Merlise Clyde <clyde@duke.edu>

See Also

`bas`

Other as.matrix methods: `list2matrix.bas()`, `which.matrix()`

Examples

```
data(Hald)
Hald.bic <-  bas.lm(Y ~ ., data=Hald, prior="BIC", initprobs="eplogp")
coef <- list2matrix.bas(Hald.bic, "mle")  # extract all ols coefficients
se <- list2matrix.bas(Hald.bic, "mle.se")
models <- list2matrix.which(Hald.bic)     #matrix of model indicators
models <- which.matrix(Hald.bic$which, Hald.bic$n.vars)     #matrix of model indicators
```

phi1 *Compound Confluent hypergeometric function of two variables*

Description

Compute the Confluent Hypergeometric function of two variables, also know as a Horn hypergeometric function or Humbert's hypergeometric used in Gordy (1998) with integral representation:

Usage

```
phi1(a, b, c, x, y, log = FALSE)
```

Arguments

a	a > 0
b	arbitrary
c	c > 0
x	x > 0
y	y > 0
log	logical indicating whether to return phi1 on the log scale

Details

phi_1(a,b,c,x,y) = [(Gamma(c)/Gamma(a) Gamma(a-c))] Int_0^1 t^(a-1) (1 - t)^(c-a-1) (1 - yt)^(-b) exp(x t) dt https://en.wikipedia.org/wiki/Humbert_series Note that Gordy's arguments for x and y are reversed in the reference above.

The original 'phi1' function in 'BAS' was based on 'C' code provided by Gordy. This function returns NA's when x is greater than 'log(.Machine$double.xmax)/2'. A more stable method for calculating the 'phi1' function using R's 'integrate' was suggested by Daniel Heemann and is now an option whenever x is too large. For calculating Bayes factors that use the 'phi1' function we recommend using the 'log=TRUE' option to compute log Bayes factors.

Author(s)

Merlise Clyde (<clyde@duke.edu>)

Daniel Heemann (<df.heemann@gmail.com>)

References

Gordy 1998

See Also

Other special functions: hypergeometric1F1(), hypergeometric2F1(), trCCH()

Examples

```
# special cases
# phi1(a, b, c, x=0, y) is the same as 2F1(b, a; c, y)
phi1(1, 2, 1.5, 0, 1 / 100, log=FALSE)
hypergeometric2F1(2, 1, 1.5, 1 / 100, log = FALSE)

# phi1(a,0,c,x,y) is the same as 1F1(a,c,x)
phi1(1, 0, 1.5, 3, 1 / 100)
hypergeometric1F1(1, 1.5, 3, log = FALSE)

# use direct integration
phi1(1, 2, 1.5, 1000, 0, log=TRUE)
```

---

plot.bas *Plot Diagnostics for an BAS Object*

Description

Four plots (selectable by 'which') are currently available: a plot of residuals against fitted values, Cumulative Model Probabilities, log marginal likelihoods versus model dimension, and marginal inclusion probabilities.

Usage

```
## S3 method for class 'bas'
plot(
  x,
  which = c(1:4),
  caption = c("Residuals vs Fitted", "Model Probabilities", "Model Complexity",
    "Inclusion Probabilities"),
  panel = if (add.smooth) panel.smooth else points,
  sub.caption = NULL,
  main = "",
  ask = prod(par("mfcol")) < length(which) && dev.interactive(),
  col.in = 2,
  col.ex = 1,
  col.pch = 1,
  cex.lab = 1,
  ...,
  id.n = 3,
  labels.id = NULL,
  cex.id = 0.75,
  add.smooth = getOption("add.smooth"),
  label.pos = c(4, 2),
  subset = NULL,
  drop.always.included = FALSE
)
```

Arguments

x	bas BMA object result of 'bas'
which	if a subset of the plots is required, specify a subset of the numbers '1:4'
caption	captions to appear above the plots
panel	panel function. The useful alternative to 'points', 'panel.smooth' can be chosen by 'add.smooth = TRUE'
sub.caption	common title-above figures if there are multiple; used as 'sub' (s.'title') otherwise. If 'NULL', as by default, a possible shortened version of deparse(x$call) is used
main	title to each plot-in addition to the above 'caption'
ask	logical; if 'TRUE', the user is asked before each plot, see 'par(ask=.)'
col.in	color for the included variables
col.ex	color for the excluded variables
col.pch	color for points in panels 1-3
cex.lab	graphics parameter to control size of variable names
...	other parameters to be passed through to plotting functions
id.n	number of points to be labeled in each plot, starting with the most extreme
labels.id	vector of labels, from which the labels for extreme points will be chosen. 'NULL' uses observation numbers
cex.id	magnification of point labels.
add.smooth	logical indicating if a smoother should be added to most plots; see also 'panel' above
label.pos	positioning of labels, for the left half and right half of the graph respectively, for plots 1-4
subset	indices of variables to include/exclude in plot of marginal posterior inclusion probabilities (NULL).
drop.always.included	logical variable to drop marginal posterior inclusion probabilities for variables that are always forced into the model. FALSE by default.

Details

This provides a panel of 4 plots: the first is a plot of the residuals versus fitted values under BMA. The second is a plot of the cumulative marginal likelihoods of models; if the model space cannot be enumerated then this provides some indication of whether the probabilities are leveling off. The third is a plot of log marginal likelihood versus model dimension and the fourth plot show the posterior marginal inclusion probabilities.

Author(s)

Merlise Clyde, based on plot.lm by John Maindonald and Martin Maechler

See Also

plot.coef.bas and image.bas.

Other bas plots: image.bas(), plot.coef.bas()

Examples

```
data(Hald)
hald.gprior =  bas.lm(Y~ ., data=Hald, prior="g-prior", alpha=13,
                      modelprior=beta.binomial(1,1),
                      initprobs="eplogp")

plot(hald.gprior)
```

---

plot.coef.bas *Plots the posterior distributions of coefficients derived from Bayesian model averaging*

Description

Displays plots of the posterior distributions of the coefficients generated by Bayesian model averaging over linear regression.

Usage

```
## S3 method for class 'coef.bas'
plot(x, e = 1e-04, subset = 1:x$n.vars, ask = TRUE, ...)
```

Arguments

x	object of class coef.bas
e	optional numeric value specifying the range over which the distributions are to be graphed.
subset	optional numerical vector specifying which variables to graph (including the intercept)
ask	Prompt for next plot
...	other parameters to be passed to plot and lines

Details

Produces plots of the posterior distributions of the coefficients under model averaging. The posterior probability that the coefficient is zero is represented by a solid line at zero, with height equal to the probability. The nonzero part of the distribution is scaled so that the maximum height is equal to the probability that the coefficient is nonzero.

The parameter e specifies the range over which the distributions are to be graphed by specifying the tail probabilities that dictate the range to plot over.

Note

For mixtures of g-priors, uncertainty in g is not incorporated at this time, thus results are approximate

Author(s)

based on function `plot.bic` by Ian Painter in package BMA; adapted for 'bas' class by Merlise Clyde <clyde@stat.duke.edu>

References

Hoeting, J.A., Raftery, A.E. and Madigan, D. (1996). A method for simultaneous variable selection and outlier identification in linear regression. Computational Statistics and Data Analysis, 22, 251-270.

See Also

`coef.bas`

Other bas plots: `image.bas()`, `plot.bas()`

Examples

```
## Not run: library(MASS)
data(UScrime)
UScrime[,-2] <- log(UScrime[,-2])
crime_bic <- bas.lm(y ~ ., data=UScrime, n.models=2^15, prior="BIC")
plot(coefficients(crime_bic), ask=TRUE)

## End(Not run)
```

---

plot.confint.bas *Plot Bayesian Confidence Intervals*

Description

Function takes the the output of functions that return credible intervals from BAS objects, and creates a plot of the posterior mean with segments representing the credible interval. of what the function does. ~~

Usage

```
## S3 method for class 'confint.bas'
plot(x, horizontal = FALSE, ...)
```

Arguments

x	the output from confint.coef.bas or confint.pred.bas containing credible intervals and estimates.
horizontal	orientation of the plot
...	optional graphical arguments to pass on to plot

Details

This function takes the HPD intervals or credible intervals created by confint.coef.bas or confint.pred.bas from BAS objects, and creates a plot of the posterior mean with segments representing the credible interval. BAS tries to return HPD intervals, and under model averaging these may not be symmetric. the description above ~~

Value

A plot of the credible intervals.

Author(s)

Merlise A Clyde

See Also

confint.coef.bas, confint.pred.bas, coef.bas, predict.bas, link{bas.lm}

Other bas methods: BAS, bas.lm(), coef.bas(), confint.coef.bas(), confint.pred.bas(), diagnostics(), fitted.bas(), force.heredity.bas(), image.bas(), predict.bas(), predict.basglm(), summary.bas(), update.bas(), variable.names.pred.bas()

Other CI methods: confint.coef.bas(), confint.pred.bas()

Examples

```
data(Hald)
hald.ZS = bas.lm(Y ~ ., data=Hald, prior="ZS-null", modelprior=uniform())
hald.coef = confint(coef(hald.ZS), parm=2:5)
plot(hald.coef)
plot(hald.coef, horizontal=TRUE)
plot(confint(predict(hald.ZS, se.fit=TRUE), parm="mean"))
```

predict.bas *Prediction Method for an object of class BAS*

Description

Predictions under model averaging or other estimators from a BMA object of class inheriting from 'bas'.

Usage

```
## S3 method for class 'bas'
predict(
  object,
  newdata,
  se.fit = FALSE,
  type = c("response", "link"),
  top = NULL,
  estimator = "BMA",
  na.action = na.pass,
  ...
)
```

Arguments

object	An object of class BAS, created by bas
newdata	dataframe for predictions. If missing, then use the dataframe used for fitting for obtaining fitted and predicted values.
se.fit	indicator for whether to compute se of fitted and predicted values
type	Type of predictions required. The defaults "reponse" is on the scale of the response is the only option currently for linear models (for Gaussian models this is equivalent to type="link").
top	a scalar integer M. If supplied, subset the top M models, based on posterior probabilities for model predictions and BMA.
estimator	estimator used for predictions. Currently supported options include: 'HPM' the highest probability model 'BMA' Bayesian model averaging, using optionally only the 'top' models 'MPM' the median probability model of Barbieri and Berger. 'BPM' the model that is closest to BMA predictions under squared error loss. BMA may be computed using only the 'top' models if supplied
na.action	function determining what should be done with missing values in newdata. The default is to predict NA.
...	optional extra arguments

Details

Use BMA and/or model selection to form predictions using the top highest probability models.

Value

a list of

`fit`	fitted values based on the selected estimator
`Ybma`	predictions using BMA, the same as fit for non-BMA methods for compatibility; will be deprecated
`Ypred`	matrix of predictions under each model for BMA
`se.fit`	se of fitted values; in the case of BMA this will be a matrix
`se.pred`	se for predicted values; in the case of BMA this will be a matrix
`se.bma.fit`	vector of posterior sd under BMA for posterior mean of the regression function. This will be NULL if estimator is not 'BMA'
`se.bma.pred`	vector of posterior sd under BMA for posterior predictive values. this will be NULL if estimator is not 'BMA'
`best`	index of top models included
`bestmodels`	subset of bestmodels used for fitting or prediction
`best.vars`	names of variables in the top model; NULL if estimator='BMA'
`df`	scalar or vector of degrees of freedom for models
`estimator`	estimator upon which 'fit' is based.

Author(s)

Merlise Clyde

See Also

`bas`, `fitted.bas`, `confint.pred.bas`, `variable.names.pred.bas`

Other predict methods: `fitted.bas()`, `predict.basglm()`, `variable.names.pred.bas()`

Other bas methods: `BAS`, `bas.lm()`, `coef.bas()`, `confint.coef.bas()`, `confint.pred.bas()`, `diagnostics()`, `fitted.bas()`, `force.heredity.bas()`, `image.bas()`, `plot.confint.bas()`, `predict.basglm()`, `summary.bas()`, `update.bas()`, `variable.names.pred.bas()`

Examples

```
data("Hald")
hald.gprior =  bas.lm(Y ~ ., data=Hald, alpha=13, prior="g-prior")

predict(hald.gprior, newdata=Hald, estimator="BPM", se.fit=TRUE)
# same as fitted
fitted(hald.gprior,estimator="BPM")
# default is BMA and estimation of mean vector
hald.bma = predict(hald.gprior, top=5, se.fit=TRUE)
confint(hald.bma)

hald.bpm = predict(hald.gprior, newdata=Hald[1,],
                   se.fit=TRUE,
                   estimator="BPM")
```

```
confint(hald.bpm)
# extract variables
variable.names(hald.bpm)

hald.hpm = predict(hald.gprior, newdata=Hald[1,],
                   se.fit=TRUE,
                   estimator="HPM")
confint(hald.hpm)
variable.names(hald.hpm)

hald.mpm = predict(hald.gprior, newdata=Hald[1,],
                   se.fit=TRUE,
                   estimator="MPM")
confint(hald.mpm)
variable.names(hald.mpm)
```

`predict.basglm` *Prediction Method for an Object of Class basglm*

Description

Predictions under model averaging from a BMA (BAS) object for GLMs under different loss functions.

Usage

```
## S3 method for class 'basglm'
predict(
  object,
  newdata,
  se.fit = FALSE,
  type = c("response", "link"),
  top = NULL,
  estimator = "BMA",
  na.action = na.pass,
  ...
)
```

Arguments

`object`	An object of class "basglm", created by `bas.glm`
`newdata`	dataframe, new matrix or vector of data for predictions. May include a column for the intercept or just the predictor variables. If a dataframe, the variables are extracted using model.matrix using the call that created 'object'. May be missing in which case the data used for fitting will be used for prediction.
`se.fit`	indicator for whether to compute se of fitted and predicted values

type	Type of predictions required. The default is "response" is on the scale of the response variable, with the alternative being on the linear predictor scale, ‘type =’link’‘. Thus for a default binomial model ‘type = ’response’‘ gives the predicted probabilities, while with ‘’link’‘, the estimates are of log-odds (probabilities on logit scale).
top	A scalar integer M. If supplied, calculate results using the subset of the top M models based on posterior probabilities.
estimator	estimator used for predictions. Currently supported options include: ’HPM’ the highest probability model ’BMA’ Bayesian model averaging, using optionally only the ’top’ models ’MPM’ the median probability model of Barbieri and Berger. ’BPM’ the model that is closest to BMA predictions under squared error loss. BMA may be computed using only the ’top’ models if supplied
na.action	function determining what should be done with missing values in newdata. The default is to predict NA.
...	optional extra arguments

Details

This function first calls the predict method for class bas (linear models) to form predictions on the linear predictor scale for ‘BMA‘, ‘HPM‘, ‘MPM‘ etc. If the estimator is ‘BMA‘ and ‘type=’response’‘ then the inverse link is applied to fitted values for type equal ‘’link’‘ and model averaging takes place in the ‘response‘ scale. Thus applying the inverse link to BMA estimate with ‘type = ’link’‘ is not equal to the fitted values for ‘type = ’response’‘ under BMA due to the nonlinear transformation under the inverse link.

Value

a list of

fit	predictions using BMA or other estimators
Ypred	matrix of predictions under model(s)
postprobs	renormalized probabilities of the top models
best	index of top models included

Author(s)

Merlise Clyde

See Also

bas.glm, predict.bas, fitted.bas

Other predict methods: fitted.bas(), predict.bas(), variable.names.pred.bas()

Other bas methods: BAS, bas.lm(), coef.bas(), confint.coef.bas(), confint.pred.bas(), diagnostics(), fitted.bas(), force.heredity.bas(), image.bas(), plot.confint.bas(), predict.bas(), summary.bas(), update.bas(), variable.names.pred.bas()

Examples

```
data(Pima.tr, package="MASS")
data(Pima.te, package="MASS")
Pima.bas = bas.glm(type ~ ., data=Pima.tr, n.models= 2^7, method="BAS",
           betaprior=CCH(a=1, b=nrow(Pima.tr)/2, s=0), family=binomial(),
           modelprior=uniform())
pred = predict(Pima.bas, newdata=Pima.te, top=1)  # Highest Probability model
cv.summary.bas(pred$fit, Pima.te$type, score="miss-class")
```

`print.bas` *Print a Summary of Bayesian Model Averaging objects from BAS*

Description

`summary` and `print` methods for Bayesian model averaging objects created by `bas` Bayesian Adaptive Sampling

Usage

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

Arguments

`x`	object of class 'bas'
`digits`	optional number specifying the number of digits to display
`...`	other parameters to be passed to `print.default`

Details

The print methods display a view similar to `print.lm` . The summary methods display a view specific to Bayesian model averaging giving the top 5 highest probability models represented by their inclusion indicators. Summaries of the models include the Bayes Factor (BF) of each model to the model with the largest marginal likelihood, the posterior probability of the models, R2, dim (which includes the intercept) and the log of the marginal likelihood.

Author(s)

Merlise Clyde <clyde@stat.duke.edu>

See Also

`coef.bas`

Examples

```
library(MASS)
data(UScrime)
UScrime[, -2] <- log(UScrime[, -2])
crime.bic <- bas.lm(y ~ ., data = UScrime, n.models = 2^15, prior = "BIC", initprobs = "eplogp")
print(crime.bic)
summary(crime.bic)
```

---

protein *Protein Activity Data*

Description

This data sets includes several predictors of protein activity from an experiment run at Glaxo.

Format

`protein` is a dataframe with 96 observations and 8 predictor variables of protein activity:

[,1]	buf	factor	Buffer
[,2]	pH	numeric	
[,3]	NaCl	numeric	
[,4]	con	numeric	protein concentration
[,5]	ra	factor	reducing agent
[,6]	det	factor	detergent
[,7]	MgCl2	numeric	
[,8]	temp	numeric	(temperature)
[,9]	prot.act1	numeric	
[,10]	prot.act2	numeric	
[,11]	prot.act3	numeric	
[,12]	prot.act4	numeric	protein activity

Source

Clyde, M. A. and Parmigiani, G. (1998), Protein Construct Storage: Bayesian Variable Selection and Prediction with Mixtures, Journal of Biopharmaceutical Statistics, 8, 431-443

robust *Robust-Prior Distribution for Coefficients in BMA Model*

Description

Creates an object representing the robust prior of Bayarri et al (2012) that is mixture of g-priors on coefficients for BAS.

Usage

```
robust(n = NULL)
```

Arguments

n	the sample size.

Details

Creates a prior structure used for bas.glm.

Value

returns an object of class "prior", with the family and hyerparameters.

Author(s)

Merlise Clyde

See Also

CCH and bas.glm

Other beta priors: CCH(), EB.local(), IC.prior(), Jeffreys(), TG(), beta.prime(), g.prior(), hyper.g(), hyper.g.n(), intrinsic(), tCCH(), testBF.prior()

Examples

```
robust(100)
```

summary.bas *Summaries of Bayesian Model Averaging objects from BAS*

Description

summary and print methods for Bayesian model averaging objects created by bas Bayesian Adaptive Sampling

Usage

```
## S3 method for class 'bas'
summary(object, n.models = 5, ...)
```

Arguments

object	object of class 'bas'
n.models	optional number specifying the number of best models to display in summary
...	other parameters to be passed to summary.default

Details

The print methods display a view similar to print.lm . The summary methods display a view specific to Bayesian model averaging giving the top 5 highest probability models represented by their inclusion indicators. Summaries of the models include the Bayes Factor (BF) of each model to the model with the largest marginal likelihood, the posterior probability of the models, R2, dim (which includes the intercept) and the log of the marginal likelihood.

Author(s)

Merlise Clyde <clyde@duke.edu>

See Also

coef.bas

Other bas methods: BAS, bas.lm(), coef.bas(), confint.coef.bas(), confint.pred.bas(), diagnostics(), fitted.bas(), force.heredity.bas(), image.bas(), plot.confint.bas(), predict.bas(), predict.basglm(), update.bas(), variable.names.pred.bas()

Examples

```
data(UScrime, package = "MASS")
UScrime[, -2] <- log(UScrime[, -2])
crime.bic <- bas.lm(y ~ ., data = UScrime, n.models = 2^15, prior = "BIC", initprobs = "eplogp")
print(crime.bic)
summary(crime.bic)
```

tCCH *Generalized tCCH g-Prior Distribution for Coefficients in BMA Models*

Description

Creates an object representing the tCCH mixture of g-priors on coefficients for BAS.

Usage

```
tCCH(alpha = 1, beta = 2, s = 0, r = 3/2, v = 1, theta = 1)
```

Arguments

alpha	a scalar > 0, recommended alpha=.5 (betaprime) or 1.
beta	a scalar > 0. The value is not updated by the data; beta should be a function of n for consistency under the null model.
s	a scalar, recommended s=0 a priori
r	r arbitrary; in the hyper-g-n prior sets r = (alpha + 2)
v	0 < v
theta	theta > 1

Details

Creates a structure used for bas.glm.

Value

returns an object of class "prior", with the family and hyerparameters.

Author(s)

Merlise Clyde

See Also

CCH, robust, hyper.g, hyper.g.nbas.glm

Other beta priors: CCH(), EB.local(), IC.prior(), Jeffreys(), TG(), beta.prime(), g.prior(), hyper.g(), hyper.g.n(), intrinsic(), robust(), testBF.prior()

Examples

```
n <- 500
tCCH(alpha = 1, beta = 2, s = 0, r = 1.5, v = 1, theta = 1 / n)
```

testBF.prior *Test based Bayes Factors for BMA Models*

Description

Creates an object representing the prior distribution on coefficients for BAS that corresponds to the test-based Bayes Factors.

Usage

```
testBF.prior(g)
```

Arguments

g	a scalar used in the covariance of Zellner's g-prior, Cov(beta) = sigma^2 g (X'X)^-

Details

Creates a prior object structure used for BAS in 'bas.glm'.

Value

returns an object of class "prior", with the family and hyerparameters.

Author(s)

Merlise Clyde

See Also

g.prior, bas.glm

Other beta priors: CCH(), EB.local(), IC.prior(), Jeffreys(), TG(), beta.prime(), g.prior(), hyper.g(), hyper.g.n(), intrinsic(), robust(), tCCH()

Examples

```
testBF.prior(100)
library(MASS)
data(Pima.tr)

# use g = n
bas.glm(type ~ .,
  data = Pima.tr, family = binomial(),
  betaprior = testBF.prior(nrow(Pima.tr)),
  modelprior = uniform(), method = "BAS"
)
```

TG *Generalized g-Prior Distribution for Coefficients in BMA Models*

Description

Creates an object representing the Truncated Gamma (tCCH) mixture of g-priors on coefficients for BAS, where u = 1/(1+g) has a Gamma distribution supported on (0, 1].

Usage

```
TG(alpha = 2)
```

Arguments

alpha	a scalar > 0, recommended alpha=.5 (betaprime) or 1. alpha=2 corresponds to the uniform prior on the shrinkage factor.

Details

Creates a structure used for bas.glm.

Value

returns an object of class "prior", with the family and hyerparameters.

Author(s)

Merlise Clyde

See Also

CCH bas.glm

Other beta priors: CCH(), EB.local(), IC.prior(), Jeffreys(), beta.prime(), g.prior(), hyper.g(), hyper.g.n(), intrinsic(), robust(), tCCH(), testBF.prior()

Examples

```
TG(alpha = 2)
CCH(alpha = 2, beta = 100, s = 0)
```

`tr.beta.binomial` *Truncated Beta-Binomial Prior Distribution for Models*

Description

Creates an object representing the prior distribution on models for BAS using a truncated Beta-Binomial Distribution on the Model Size

Usage

```
tr.beta.binomial(alpha = 1, beta = 1, trunc)
```

Arguments

`alpha`	parameter in the beta prior distribution
`beta`	parameter in the beta prior distribution
`trunc`	parameter that determines truncation in the distribution i.e. P(M; alpha, beta, trunc) = 0 if M > trunc.

Details

The beta-binomial distribution on model size is obtained by assigning each variable inclusion indicator independent Bernoulli distributions with probability w, and then giving w a beta(alpha,beta) distribution. Marginalizing over w leads to the number of included predictors having a beta-binomial distribution. The default hyperparameters lead to a uniform distribution over model size. The Truncated version assigns zero probability to all models of size > trunc.

Value

returns an object of class "prior", with the family and hyperparameters.

Author(s)

Merlise Clyde

See Also

`bas.lm`, `Bernoulli`,`uniform`

Other priors modelpriors: `Bernoulli()`, `Bernoulli.heredity()`, `beta.binomial()`, `tr.poisson()`, `tr.power.prior()`, `uniform()`

Examples

```
tr.beta.binomial(1, 10, 5)
library(MASS)
data(UScrime)
UScrime[, -2] <- log(UScrime[, -2])
crime.bic <- bas.lm(y ~ .,
  data = UScrime, n.models = 2^15, prior = "BIC",
  modelprior = tr.beta.binomial(1, 1, 8),
  initprobs = "eplogp"
)
```

---

tr.poisson *Truncated Poisson Prior Distribution for Models*

Description

Creates an object representing the prior distribution on models for BAS using a truncated Poisson Distribution on the Model Size

Usage

```
tr.poisson(lambda, trunc)
```

Arguments

lambda	parameter in the Poisson distribution representing expected model size with infinite predictors
trunc	parameter that determines truncation in the distribution i.e. P(M; lambda, trunc) = 0 if M > trunc

Details

The Poisson prior distribution on model size is obtained by assigning each variable inclusion indicator independent Bernoulli distributions with probability w, and then taking a limit as p goes to infinity and w goes to zero, such that p*w converges to lambda. The Truncated version assigns zero probability to all models of size M > trunc.

Value

returns an object of class "prior", with the family and hyperparameters.

Author(s)

Merlise Clyde

See Also

bas.lm, Bernoulli,uniform

Other priors modelpriors: Bernoulli(), Bernoulli.heredity(), beta.binomial(), tr.beta.binomial(), tr.power.prior(), uniform()

Examples

```
tr.poisson(10, 50)
```

---

tr.power.prior *Truncated Power Prior Distribution for Models*

Description

Creates an object representing the prior distribution on models for BAS using a truncated Distribution on the Model Size where the probability of gamma proportional to p^-kappa |gamma| where gamma is the vector of model indicators and |gamma| is the model size.

Usage

```
tr.power.prior(kappa = 2, trunc)
```

Arguments

kappa	parameter in the prior distribution that controls sparsity
trunc	parameter that determines truncation in the distribution i.e. P(gamma; alpha, beta, trunc) = 0 if |gamma| > trunc.

Details

The Truncated version assigns zero probability to all models of size > trunc.

Value

returns an object of class "prior", with the family and hyperparameters.

Author(s)

Merlise Clyde

See Also

bas.lm, Bernoulli,uniform

Other priors modelpriors: Bernoulli(), Bernoulli.heredity(), beta.binomial(), tr.beta.binomial(), tr.poisson(), uniform()

Examples

```
tr.power.prior(2, 8)
library(MASS)
data(UScrime)
UScrime[, -2] <- log(UScrime[, -2])
crime.bic <- bas.lm(y ~ .,
  data = UScrime, n.models = 2^15, prior = "BIC",
  modelprior = tr.power.prior(2, 8),
  initprobs = "eplogp"
)
```

---

trCCH *Truncated Compound Confluent Hypergeometric function*

Description

Compute the Truncated Confluent Hypergeometric function from Li and Clyde (2018) which is the normalizing constant in the tcch density of Gordy (1998) with integral representation:

Usage

```
trCCH(a, b, r, s, v, k, log = FALSE)
```

Arguments

a	a > 0
b	b > 0
r	r >= 0
s	arbitrary
v	0 < v
k	arbitrary
log	logical indicating whether to return values on the log scale; useful for Bayes Factor calculations

Details

tr.cch(a,b,r,s,v,k) = Int_0^1/v u^(a-1) (1 - vu)^(b -1) (k + (1 - k)vu)^(-r) exp(-s u) du

This uses a more stable method for calculating the normalizing constant using R's 'integrate' function rather than the version in Gordy 1998. For calculating Bayes factors that use the 'trCCH' function we recommend using the 'log=TRUE' option to compute log Bayes factors.

Author(s)

Merlise Clyde (<clyde@duke.edu>)

References

Gordy 1998 Li & Clyde 2018

See Also

Other special functions: hypergeometric1F1(), hypergeometric2F1(), phi1()

Examples

```
# special cases
# trCCH(a, b, r, s=0, v = 1, k) is the same as
# 2F1(a, r, a + b, 1 - 1/k)*beta(a, b)/k^r

k = 10; a = 1.5; b = 2; r = 2;
trCCH(a, b, r, s=0, v = 1, k=k) *k^r/beta(a,b)
hypergeometric2F1(a, r, a + b, 1 - 1/k, log = FALSE)

# trCCH(a,b,0,s,1,1) is the same as
# beta(a, b) 1F1(a, a + b, -s, log=FALSE)
s = 3; r = 0; v = 1; k = 1
beta(a, b)*hypergeometric1F1(a, a+b, -s, log = FALSE)
trCCH(a, b, r, s, v, k)

# Equivalence with the Phi1 function
a = 1.5; b = 3; k = 1.25; s = 400;  r = 2;  v = 1;

phi1(a, r,  a + b, -s, 1 - 1/k,  log=FALSE)*(k^-r)*gamma(a)*gamma(b)/gamma(a+b)
trCCH(a,b,r,s,v,k)
```

---

uniform *Uniform Prior Distribution for Models*

Description

Creates an object representing the prior distribution on models for BAS.

Usage

```
uniform()
```

Details

The Uniform prior distribution is a commonly used prior in BMA, and is a special case of the independent Bernoulli prior with probs=.5. The implied prior distribution on model size is binomial(p, .5).

Value

returns an object of class "prior", with the family name Uniform.

Author(s)

Merlise Clyde

See Also

bas.lm, beta.binomial,Bernoulli,

Other priors modelpriors: Bernoulli(), Bernoulli.heredity(), beta.binomial(), tr.beta.binomial(), tr.poisson(), tr.power.prior()

Examples

```
uniform()
```

---

update.bas *Update BAS object using a new prior*

Description

Update a BMA object using a new prior distribution on the coefficients.

Usage

```
## S3 method for class 'bas'
update(object, newprior, alpha = NULL, ...)
```

Arguments

object	BMA object to update
newprior	Update posterior model probabilities, probne0, shrinkage, logmarg, etc, using prior based on newprior. See bas for available methods
alpha	optional new value of hyperparameter in prior for method
...	optional arguments

Details

Recomputes the marginal likelihoods for the new methods for models already sampled in current object.

Value

A new object of class BMA

Author(s)

Merlise Clyde <clyde@stat.duke.edu>

References

Clyde, M. Ghosh, J. and Littman, M. (2010) Bayesian Adaptive Sampling for Variable Selection and Model Averaging. Journal of Computational Graphics and Statistics. 20:80-101 doi:10.1198/jcgs.2010.09049

See Also

bas for available methods and choices of alpha

Other bas methods: BAS, bas.lm(), coef.bas(), confint.coef.bas(), confint.pred.bas(), diagnostics(), fitted.bas(), force.heredity.bas(), image.bas(), plot.confint.bas(), predict.bas(), predict.basglm(), summary.bas(), variable.names.pred.bas()

Examples

```
library(MASS)
data(UScrime)
UScrime[,-2] <- log(UScrime[,-2])
crime.bic <-  bas.lm(y ~ ., data=UScrime, n.models=2^10, prior="BIC",initprobs= "eplogp")
crime.ebg <- update(crime.bic, newprior="EB-global")
crime.zs <- update(crime.bic, newprior="ZS-null")
```

---

variable.names.pred.bas

Extract the variable names for a model from a BAS prediction object

---

Description

S3 method for class 'pred.bas'. Simple utility function to extract the variable names. Used to print names for the selected models using estimators for 'HPM', 'MPM' or 'BPM". for the selected model created by predict for BAS objects.

Usage

```
## S3 method for class 'pred.bas'
variable.names(object, ...)
```

Arguments

object	a BAS object created by predict from a BAS 'bas.lm' or 'bas.glm' object
...	other arguments to pass on

Value

a character vector with the names of the variables included in the selected model; in the case of 'BMA' this will be all variables

See Also

predict.bas

Other predict methods: fitted.bas(), predict.bas(), predict.basglm()

Other bas methods: BAS, bas.lm(), coef.bas(), confint.coef.bas(), confint.pred.bas(), diagnostics(), fitted.bas(), force.heredity.bas(), image.bas(), plot.confint.bas(), predict.bas(), predict.basglm(), summary.bas(), update.bas()

Examples

```
data(Hald)
hald.gprior =  bas.lm(Y~ ., data=Hald, prior="ZS-null", modelprior=uniform())
hald.bpm = predict(hald.gprior, newdata=Hald[1,],
                   se.fit=TRUE,
                   estimator="BPM")
variable.names(hald.bpm)
```

---

which.matrix *Coerce a BAS list object of models into a matrix.*

Description

This function coerces the list object of models to a matrix and fill in the zeros to facilitate other computations.

Usage

```
which.matrix(which, n.vars)
```

Arguments

which	a 'bas' model object x$which
n.vars	the total number of predictors, x$n.vars

Details

which.matrix coerces x$which into a matrix.

Value

a matrix representation of x$which, with number of rows equal to the length of which.models or total number of models and number of columns x$n.vars

Author(s)

Merlise Clyde <clyde@duke.edu>

See Also

bas

Other as.matrix methods: list2matrix.bas(), list2matrix.which()

Examples

```
data(Hald)
Hald.bic <-  bas.lm(Y ~ ., data=Hald, prior="BIC", initprobs="eplogp")
# matrix of model indicators
models <- which.matrix(Hald.bic$which, Hald.bic$n.vars)
```

Index

∗ **BAS methods**
bas.lm, 10
∗ **BMA functions**
bas.glm, 5
∗ **BMA**
bas.glm, 5
bas.lm, 10
∗ **CI methods**
confint.coef.bas, 32
confint.pred.bas, 33
plot.confint.bas, 60
∗ **GLM**
bas.glm, 5
bayesglm.fit, 21
∗ **as.matrix methods**
list2matrix.bas, 54
list2matrix.which, 55
which.matrix, 80
∗ **bas coefs**
coef.bas, 30
∗ **bas methods**
BAS, 3
bas.lm, 10
coef.bas, 30
confint.coef.bas, 32
confint.pred.bas, 33
diagnostics, 35
fitted.bas, 41
force.heredity.bas, 43
image.bas, 50
plot.confint.bas, 60
predict.bas, 62
predict.basglm, 64
summary.bas, 69
update.bas, 78
variable.names.pred.bas, 79
∗ **bas plots**
image.bas, 50
plot.bas, 57
plot.coef.bas, 59
∗ **bayesian**
plot.confint.bas, 60
∗ **beta priors**
beta.prime, 25
CCH, 28
EB.local, 38
g.prior, 44
hyper.g, 45
hyper.g.n, 46
IC.prior, 49
intrinsic, 52
Jeffreys, 53
robust, 68
tCCH, 70
testBF.prior, 71
TG, 72
∗ **coef priors**
EB.global, 36
∗ **datasets**
bodyfat, 26
Hald, 45
protein, 67
∗ **math**
hypergeometric1F1, 47
hypergeometric2F1, 48
phi1, 56
trCCH, 76
∗ **package**
BAS, 3
∗ **predict methods**
fitted.bas, 41
predict.bas, 62
predict.basglm, 64
variable.names.pred.bas, 79
∗ **print**
print.bas, 66
summary.bas, 69
∗ **priors modelpriors**

Bernoulli, 23
Bernoulli.heredity, 24
beta.binomial, 24
tr.beta.binomial, 73
tr.poisson, 74
tr.power.prior, 75
uniform, 77
∗ **regression**
BAS, 3
bas.glm, 5
bas.lm, 10
bayesglm.fit, 21
coef.bas, 30
confint.coef.bas, 32
confint.pred.bas, 33
cv.summary.bas, 34
EB.global, 36
eplogprob, 38
eplogprob.marg, 40
fitted.bas, 41
force.heredity.bas, 43
image.bas, 50
list2matrix.bas, 54
list2matrix.which, 55
plot.bas, 57
plot.coef.bas, 59
plot.confint.bas, 60
predict.bas, 62
predict.basglm, 64
print.bas, 66
summary.bas, 69
update.bas, 78
which.matrix, 80
∗ **special functions**
hypergeometric1F1, 47
hypergeometric2F1, 48
phi1, 56
trCCH, 76
∗ **variable selection**
bas.glm, 5
bas.lm, 10

AIC, *6*
AIC.prior *(*IC.prior*)*, 49
aic.prior *(*IC.prior*)*, 49

BAS, 3, *18*, *31*, *33, 34*, *36*, *42, 43*, *51*, *61*, *63*, *65*, *69*, *79, 80*
bas, *31*, *37*, *39*, *41*, *50, 51*, *54, 55*, *63*, *78, 79*, *81*
bas *(*bas.lm*)*, 10
bas.glm, *4*, 5, *25*, *28, 29*, *38*, *46, 47*, *52, 53*, *65*, *68*, *70–72*
bas.lm, *4*, 10, *23*, *25*, *31*, *33, 34*, *36*, *42, 43*, *51*, *61*, *63*, *65*, *69*, *73*, *75*, *78–80*
Bayes.outlier, 20
bayesglm.fit, 21
Bernoulli, *6*, *13*, 23, *24, 25*, *73*, *75*, *78*
bernoulli *(*Bernoulli*)*, 23
Bernoulli.heredity, *23*, 24, *25*, *73*, *75*, *78*
Beta.Binomial *(*beta.binomial*)*, 24
beta.binomial, *6*, *13*, *23*, 24, *24*, *73*, *75*, *78*
beta.prime, *6*, 25, *29*, *38*, *45–47*, *50*, *52, 53*, *68*, *70–72*
BIC, *6*
BIC.prior *(*IC.prior*)*, 49
bic.prior, *22*, *29*
bic.prior *(*IC.prior*)*, 49
Bodyfat *(*bodyfat*)*, 26
bodyfat, 26

CCH, *6*, *26*, 28, *38*, *45–47*, *50*, *52, 53*, *68*, *70–72*
climate, 29
coef *(*coef.bas*)*, 30
coef.bas, *4*, *18*, 30, *33, 34*, *36*, *42, 43*, *51*, *60, 61*, *63*, *65, 66*, *69*, *79, 80*
coefficients *(*coef.bas*)*, 30
coefficients.bas, *17, 18*
confint *(*confint.coef.bas*)*, 32
confint.coef.bas, *4*, *18*, *31*, 32, *34*, *36*, *42, 43*, *51*, *61*, *63*, *65*, *69*, *79, 80*
confint.pred.bas, *4*, *18*, *31*, 33, *33*, *36*, *42, 43*, *51*, *61*, *63*, *65*, *69*, *79, 80*
cv.summary.bas, 34

diagnostics, *4*, *14*, *17, 18*, *31*, *33, 34*, 35, *42, 43*, *51*, *61*, *63*, *65*, *69*, *79, 80*

EB *(*EB.local*)*, 38
EB.global, 36
EB.local, *6*, *26*, *29*, 38, *45–47*, *50*, *52, 53*, *68*, *70–72*
eplogprob, *6*, *13*, *18*, 38
eplogprob.marg, *6*, *13*, 40

fitted *(*fitted.bas*)*, 41
fitted.bas, *4*, *17, 18*, *31*, *33, 34*, *36*, 41, *43*, *51*, *61*, *63*, *65*, *69*, *79, 80*

force.heredity.bas, *4*, *18*, *31*, *33, 34*, *36*, *42*, 43, *51*, *61*, *63*, *65*, *69*, *79, 80*

g.prior, *6*, *26*, *29*, *38*, 44, *46, 47*, *50*, *52, 53*, *68*, *70–72*
glm, *22*
glm.fit, *22*

Hald, 45
hald *(*Hald*)*, 45
hyper.g, *26*, *29*, *38*, 45, *45*, *47*, *50*, *52, 53*, *68*, *70–72*
hyper.g.n, *26*, *29*, *38*, *45*, 46, *46*, *50*, *52, 53*, *68*, *70–72*
hypergeometric1F1, 47, *49*, *56*, *77*
hypergeometric2F1, 48, *48*, *56*, *77*

IC.prior, *26*, *29*, *38*, *45–47*, 49, *52, 53*, *68*, *70–72*
image *(*image.bas*)*, 50
image.bas, *4*, *17, 18*, *31*, *33, 34*, *36*, *42, 43*, 50, *59–61*, *63*, *65*, *69*, *79, 80*
intrinsic, *6*, *26*, *29*, *38*, *45–47*, *50*, 52, *53*, *68*, *70–72*

Jeffreys, *26*, *29*, *38*, *45–47*, *50*, *52*, 53, *68*, *70–72*

list2matrix *(*list2matrix.bas*)*, 54
list2matrix.bas, 54, *55*, *81*
list2matrix.which, *54*, 55, *81*

phi1, *48, 49*, 56, *77*
plot.bas, *17, 18*, *51*, 57, *60*
plot.coef.bas, *31*, *51*, 59, *59*
plot.confint.bas, *4*, *18*, *31*, *33, 34*, *36*, *42, 43*, *51*, 60, *63*, *65*, *69*, *79, 80*
predict *(*predict.bas*)*, 62
predict.bas, *4*, *17, 18*, *31*, *33–36*, *42, 43*, *51*, *61*, 62, *65*, *69*, *79, 80*
predict.basglm, *4*, *18*, *31*, *33, 34*, *36*, *42, 43*, *51*, *61*, *63*, 64, *69*, *79, 80*
print *(*print.bas*)*, 66
print.bas, *18*, 66
print.coef.bas *(*coef.bas*)*, 30
protein, 67

robust, *6*, *26*, *29*, *38*, *45–47*, *50*, *52, 53*, 68, *70–72*

summary *(*summary.bas*)*, 69
summary.bas, *4*, *17, 18*, *31*, *33, 34*, *36*, *42, 43*, *51*, *61*, *63*, *65*, 69, *79, 80*

tCCH, *26*, *29*, *38*, *45–47*, *50*, *52, 53*, *68*, 70, *71, 72*
testBF.prior, *26*, *29*, *38*, *45–47*, *50*, *52, 53*, *68*, *70*, 71, *72*
TG, *26*, *29*, *38*, *45–47*, *50*, *52, 53*, *68*, *70, 71*, 72
tr.Beta.Binomial *(*tr.beta.binomial*)*, 73
tr.beta.binomial, *6*, *13*, *23–25*, 73, *75*, *78*
tr.Poisson *(*tr.poisson*)*, 74
tr.poisson, *13*, *23–25*, *73*, 74, *75*, *78*
tr.Power.Prior *(*tr.power.prior*)*, 75
tr.power.prior, *6*, *13*, *23–25*, *73*, 75, *75*, *78*
trCCH, *48, 49*, *56*, 76
trunc.CCH *(*trCCH*)*, 76

Uniform *(*uniform*)*, 77
uniform, *6*, *13*, *23–25*, *73*, *75*, 77
update, *37*
update *(*update.bas*)*, 78
update.bas, *4*, *17, 18*, *31*, *33, 34*, *36*, *42, 43*, *51*, *61*, *63*, *65*, *69*, 78, *80*

variable.names *(*variable.names.pred.bas*)*, 79
variable.names.pred.bas, *4*, *18*, *31*, *33, 34*, *36*, *42, 43*, *51*, *61*, *63*, *65*, *69*, 79, *79*

which.matrix, *54, 55*, 80