Package 'glmbayes'

August 3, 2025

Type Package

Title Bayesian Generalized Linear Models (iid Samples)

Version 0.1.0

Date 2025-07-24

Author Kjell Nygren

Maintainer Kjell Nygren <kjell.a.nygren@gmail.com>

Description Generates iid samples for Bayesian Generalized Linear Models.

License GPL-2

Imports stats, Rcpp (>= 1.0.4.6), RcppArmadillo, RcppParallel

LinkingTo Rcpp, RcppArmadillo, RcppParallel

Depends MASS, R (>= 3.5.0)

Suggests knitr, rmarkdown

SystemRequirements Optional OpenCL support. If available, GPU acceleration will be used; otherwise, computation runs on CPU.

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation yes

Archs x64

Contents

glmbayes-package 2
AMI 3
anova.glmb 3
carinsca 5
case.names.glmb 6
confint.glmb 7
deviance.rglmb 8
dummy.coef.glmb 9
EnvelopeBuild 10
EnvelopeOpt 13
EnvelopeSort 15
extractAIC.glmb 19

extractAIC.rglmb 20
f2_gaussian_vector 21
formula.summary.rglmb 22
glmb 23
glmb.influence.measures 28
glmb.wfit 31
glmbayes 34
glmbfamfunc 37
glmb_Standardize_Model 38
influence.glmb 41
Inv_f3_gaussian 43
lmb 45
logLik.glmb 49
menarche2 50
Neg_logLik 51
Normal_ct 51
pfamily 52
predict.glmb 55
Prior_Check 57
Prior_Setup 60
residuals.glmb 61
rGamma_reg 63
rglmb 66
rindependent_norm_gamma_reg 69
rindep_norm_gamma_reg_std_R 73
rlmb 76
rnnorm_reg_std 79
rnnorm_reg_std_cpp_parallel 83
rNormal_Gamma_reg 84
rNormal_reg 86
rNormal_reg.wfit 89
setlogP 91
Set_Grid 92
simulate.glmb 93
summary.glmb 95
summary.rglmb 96
vcov.glmb 98

Index **100**

`glmbayes-package` *glmbayes: Bayesian Generalized Linear Models (iid Samples)*

Description

Generates iid samples for Bayesian Generalized Linear Models.

AMI *amitriptyline overdose data*

Description

Data with information on 17 overdoses of the drug amitriptyline

Usage

```
data(AMI)
```

Format

This data frame contains the following columns:

`TOT` total TCAD plasma level

`AMI` amount of amitriptyline present in the TCAD plasma level

`GEN` gender (male = 0, female = 1)

`AMT` amount of drug taken at time of overdose

`PR` PR wave measurement

`DIAP` diastolic blood pressure

`QRS` QRS wave measurement

Details

Details to be added...

References

UVA Example - Add Reference Here.

Examples

```
data(menarche2)
```

anova.glmb *Analysis of Deviance for Bayesian Generalized Linear Model Fits*

Description

Compute an analysis of deviance table for one (current implementation) or more (future) Bayesian generalized linear model fits,

Usage

```
## S3 method for class 'glmb'
anova(object, ...)
```

Arguments

object	an object of class glmb, typically the result of a call to glmb
...	Other arguments passed to or from other methods.

Details

Specifying a single object (currently only implementation) gives a sequential analysis of deviance table for that fit. That is the reductions in residual deviance as each term of the formula is addded in turn are given as the rows of a table, plus the residual deviances themselves.

Value

An object of class "anova" inheriting from class "data.frame".

Examples

```
data(menarche2)
## ----Analysis Setup------------------------------------------------------------
## Number of variables in model
Age=menarche2$Age
nvars=2
## Reference Ages for setting of priors and Age_Difference
ref_age1=13  # user can modify this
ref_age2=15  ## user can modify this
## Define variables used later in analysis
Age2=menarche2$Age-ref_age1
Age_Diff=ref_age2-ref_age1
mu1=as.matrix(c(0,1.098612),ncol=1)
V1<-1*diag(nvars)
V1[1,1]=0.18687882
V1[2,2]=0.10576217
V1[1,2]=-0.03389182
V1[2,1]=-0.03389182
Menarche_Model_Data=data.frame(Age=menarche2$Age,Total=menarche2$Total,
                               Menarche=menarche2$Menarche,Age2)
glmb.out1<-glmb(n=1000,cbind(Menarche, Total-Menarche) ~Age2,family=binomial(logit),
                pfamily=dNormal(mu=mu1,Sigma=V1),data=Menarche_Model_Data)

# Prediction from original model
pred1=predict(glmb.out1,type="response")

## Get Original Residuals, their means, and credible bounds
res_out=residuals(glmb.out1)
colMeans(res_out)

## Set up to simulate new data and residuals
res_mean=colMeans(res_out)
res_low1=apply(res_out,2,FUN=quantile,probs=c(0.025))
res_high1=apply(res_out,2,FUN=quantile,probs=c(0.975))

## Simulate new data and get residuals for simulated data

ysim1=simulate(glmb.out1,nsim=1,seed=NULL,pred=pred1,family="binomial",
               prior.weights=weights(glmb.out1))
```

```
res_ysim_out1=residuals(glmb.out1,ysim=ysim1)
res_low=apply(res_ysim_out1,2,FUN=quantile,probs=c(0.025))
res_high=apply(res_ysim_out1,2,FUN=quantile,probs=c(0.975))

# Plot Credible Interval bounds for Deviance Residuals

plot(res_mean~Age,ylim=c(-2.5,2.5),
main="Credible Interval Bound for Menarche - Logit Model Deviance Residuals",
xlab = "Age", ylab = "Avg. Dev. Res")
lines(Age, 0*res_mean,lty=1)
lines(Age, res_low,lty=1)
lines(Age, res_high,lty=1)
lines(Age, res_low1,lty=2)
lines(Age, res_high1,lty=2)
```

`carinsca` *Canadian Automobile Insurance Claims for 1957-1958*

Description

The data give the Canadian automobile insurance experience for policy years 1956 and 1957 as of June 30, 1959. The data includes virtually every insurance company operating in Canada and was collated by the Statistical Agency (Canadian Underwriters' Association - Statistical Department) acting under instructions from the Superintendent of Insurance. The data given here is for private passenger automobile liability for non-farmers for all of Canada excluding Saskatchewan.

Usage

```
data(carinsca)
```

Format

A data frame with 20 observations on the following 6 variables.

`Merit` Merit Rating:
- 3 - licensed and accident free 3 or more years
- 2 - licensed and accident free 2 years
- 1 - licensed and accident free 1 year
- 0 - all others

`Class` 1 - pleasure, no male operator under 25
- 2 - pleasure, non-principal male operator under 25
- 3 - business use
- 4 - unmarried owner or principal operator under 25
- 5 - married owner or principal operator under 25

`Insured` Earned car years

`Premium` Earned premium in 1000's
(adjusted to what the premium would have been had all cars been written at 01 rates)

`Claims` Number of claims

`Cost` Total cost of the claim in 1000's of dollars

Details

One could apply Poisson regression to the number of claims and gamma regression to the cost per claim.

Source

Bailey, R. A., and Simon, LeRoy J. (1960). Two studies in automobile insurance ratemaking. ASTIN Bulletin, 192-217.

References

Data downloaded from http://www.statsci.org/data/general/carinsca.html.
That site also contains classical Poisson and Gamma regression examples.

Examples

```
data(carinsca)
## maybe str(carinsca) ; plot(carinsca) ...
```

`case.names.glmb` *Case and Variable Names of Fitted Models*

Description

Simple utilities returning (non-missing) case names, and (non-eliminated) variable names.

Usage

```
## S3 method for class 'glmb'
case.names(object, full = FALSE, ...)

## S3 method for class 'glmb'
variable.names(object, full = FALSE, ...)
```

Arguments

`object`	a fitted model object, typically of class `"glmb"`.
`full`	logical; if `TRUE`, all names (including zero weights, ...) are returned.
`...`	further arguments passed to or from other methods.

Value

A character vector

Examples

```
## ----dobson----------------------------------------------------------------
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)

## Prior mean vector
mu<-matrix(0,5)
mu[1,1]=log(mean(counts))
## Prior standard deviation and Variance
mysd<-1
V=((mysd)^2)*diag(5)
## Call to glmb
glmb.D93<-glmb(n=1000,counts ~ outcome + treatment,
               family = poisson(),pfamily=dNormal(mu=mu,Sigma=V))

## ----glmb vcov-------------------------------------------------------------
case.names(glmb.D93)
```

`confint.glmb` *Credible Intervals for Model Parameters*

Description

Computes credible intervals for Model Parameters

Usage

```
## S3 method for class 'glmb'
confint(object, parm, level = 0.95, ...)
```

Arguments

`object`	a fitted model object of class `"glmb"`. Typically the result of a call to glmb.
`parm`	a specification (not yet implemented) of which parameters are to be given credible sets, either a vector of numbers or a vector of names. If missing, all parameters are considered.
`level`	the credible interval required.
`...`	additional argument(s) for methods.

Value

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

Examples

```
## ----dobson----------------------------------------------------------------
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)

## Prior mean vector
mu<-matrix(0,5)
mu[1,1]=log(mean(counts))
## Prior standard deviation and Variance
mysd<-1
V=((mysd)^2)*diag(5)
## Call to glmb
glmb.D93<-glmb(n=1000,counts ~ outcome + treatment,
               family = poisson(),pfamily=dNormal(mu=mu,Sigma=V))
## ----glmb confint----------------------------------------------------------
confint(glmb.D93)
```

`deviance.rglmb` *Model Deviance*

Description

Returns the deviance of a fitted Bayesian Generalized Linear Model

Usage

```
## S3 method for class 'rglmb'
deviance(object, ...)
```

Arguments

`object`	an object of class `"rglmb"`, typically the result of a call to rglmb
`...`	further arguments to or from other methods

Value

A vector with the deviance extracted from the `object`.

Examples

```
## ----dobson----------------------------------------------------------------
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)

## Prior mean vector
mu<-matrix(0,5)
mu[1,1]=log(mean(counts))
## Prior standard deviation and Variance
mysd<-1
```

```
V=((mysd)^2)*diag(5)
## Call to glmb
glmb.D93<-glmb(n=1000,counts ~ outcome + treatment,
               family = poisson(),pfamily=dNormal(mu=mu,Sigma=V))
## ----glmb extractAIC-------------------------------------------------------
extractAIC(glmb.D93)
```

`dummy.coef.glmb` *Extract Coefficients in Original Coding*

Description

This extracts coefficients in terms of the original levels of the coefficients rather than the coded variable

Usage

```
## S3 method for class 'glmb'
dummy.coef(object, use.na = FALSE, ...)

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

Arguments

`object`	a `glmb` model fit
`use.na`	logical flag for coefficients in a singular model. If use.na is true, undetermined coefficients will be missing; if false they will get one possible value.
`x`	object to be printed
`...`	arguments passed to or from other methods

Details

A fitted linear model has coefficients for the contrasts of the factor terms, usually one less in number than the number of levels. This function re-expresses the coefficients in the original coding; as the coefficients will have been fitted in the reduced basis, any implied constraints (e.g., zero sum for contr.helmert or contr.sum) will be respected. There will be little point in using dummy.coef for contr.treatment contrasts, as the missing coefficients are by definition zero.

Value

A list giving for each term the draws for the coefficients.

Examples

```
set.seed(333)
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
mysd<-1
mu<-matrix(0,5)
```

```
mu[1]=log(mean(counts))
V0<-((mysd)^2)*diag(5)
glmb.D93<-glmb(n=1000,counts ~ outcome + treatment,
family = poisson(),pfamily=dNormal(mu=mu,Sigma=V0))
summary(glmb.D93)


myd=dummy.coef(glmb.D93)
print(myd)
```

`EnvelopeBuild` *Builds Envelope function for simulation*

Description

Builds an Enveloping function for simulation using a grid and tangencies for the posterior density

Usage

```
EnvelopeBuild(
  bStar,
  A,
  y,
  x,
  mu,
  P,
  alpha,
  wt,
  family = "binomial",
  link = "logit",
  Gridtype = 2L,
  n = 1L,
  sortgrid = FALSE
)
```

Arguments

`bStar`	Point at which envelope should be centered (typically posterior mode
`A`	Diagonal precision matrix for log-likelihood function associated with model in standard form
`y`	a vector of observations of length `m`
`x`	a design matrix of dimension `m * p`
`mu`	a vector giving the prior means of the variables
`P`	a positive-definite symmetric matrix specifying the prior precision matrix of the variables
`alpha`	offset vector
`wt`	a vector of weights
`family`	Family for the envelope. Can take on values binomial, quasibinomial, poisson, quasibinomial, and Gamma

link	Link function for the envelope. Valid links for the binomial and quasibinomial families are logit, probit, and cloglog. The valid value for the poisson, quasipoisson, and the Gamma families is the log
Gridtype	an optional argument specifying the method used to determine number of likelihood subgradient densities used to construct the enveloping function
n	number of draws to generate from posterior density. Used here to help determine the size of the grid (Gridtype 1 and 2 only)
sortgrid	an optional Logical argument determining whether the final envelope should be sorted in descending order based on probability for each part of the envelope

Details

To construct an enveloping function, we follow the approach in Nygren and Nygren (2006) which involves the following steps when a maximally sized grid is constructed (if the prior for some dimensions is relatively strong, this may not be needed)

1) For each dimension, a constant `omega_i` is found that depends on the corresponding diagonal element in the precision matrix.

2) Corresponding intervals `(thetastar_i-0.5*omega,thetastar_i-0.5*omega)` are constructed around the posterior mode thetastar for each dimension

3) The mode as well as the points `thetastar_i-omega_i` and `thetastar_i+omega_i` are selected as the components of the points at which tangencies will be found for each of the dimensions.

4) A Grid is constructed with all possible combinations of points and negative log-likelihood and gradient for the negative log-likelihood are evaluated (see the EnvelopeBuild_c.cpp function source code for details)

5) The `Set_Grid` function is called in order to evaluate the log of the density associated with each of the resulting restricted multivariate normals by evaluating the differences between the cummulative density for each dimension between its lower and upper bound.

6) The Set_LogP is called in order to help set the probabilities with which each of the components of the grid should be sampled (see remark 6 in Nygren and Nygren (2006)).

Any constants needed by the sampling are added to a list and returned by the function.

Value

The function returns a list consisting of the following components (the first six of which are matrices with the number of rows equal to the number of components in the Grid and columns equal to the number of parameters):

GridIndex	A matrix containing information on how each dimension should be sampled (1 means left tail of a restricted normal, 2 center, 3 right tail, and 4 the entire line)
thetabars	A matrix containing the points of tangencies associated with each component of the grid
cbars	A matrix containing the gradients for the negative log-likelihood at each tangency
logU	A matrix containing the log of the cummulative probability associated with each dimension
logrt	A matrix containing the log of the probability associated with the right tail (i.e. that to the right of the lower bound)
loglt	A matrix containing the log of the probability associated with the left tail (i.e., that to the left of the upper bound)

LLconst	A vector containing constant for each component of the grid used during the accept-reject procedure
logP	A matrix containing log-probabilities related to the components of the grid
PLSD	A vector containing the probability of each component in the Grid

Examples

```
data(menarche2)

summary(menarche2)
plot(Menarche/Total ~ Age, data=menarche2)

Age2=menarche2$Age-13

x<-matrix(as.numeric(1.0),nrow=length(Age2),ncol=2)
x[,2]=Age2

y=menarche2$Menarche/menarche2$Total
wt=menarche2$Total

mu<-matrix(as.numeric(0.0),nrow=2,ncol=1)
mu[2,1]=(log(0.9/0.1)-log(0.5/0.5))/3

V1<-1*diag(as.numeric(2.0))

# 2 standard deviations for prior estimate at age 13 between 0.1 and 0.9
## Specifies uncertainty around the point estimates

V1[1,1]<-((log(0.9/0.1)-log(0.5/0.5))/2)^2
V1[2,2]=(3*mu[2,1]/2)^2  # Allows slope to be up to 1 times as large as point estimate

famfunc<-glmbfamfunc(binomial(logit))

f1<-famfunc$f1
f2<-famfunc$f2
f3<-famfunc$f3
f5<-famfunc$f5
f6<-famfunc$f6

dispersion2<-as.numeric(1.0)
start <- mu
offset2=rep(as.numeric(0.0),length(y))
P=solve(V1)
n=1000


###### Adjust weight for dispersion

wt2=wt/dispersion2

######################### Shift mean vector to offset so that adjusted model has 0 mean

alpha=x%*%as.vector(mu)+offset2
mu2=0*as.vector(mu)
P2=P
x2=x
```

```
#####  Optimization step to find posterior mode and associated Precision

parin=start-mu

opt_out=optim(parin,f2,f3,y=as.vector(y),x=as.matrix(x),mu=as.vector(mu2),
              P=as.matrix(P),alpha=as.vector(alpha),wt=as.vector(wt2),
              method="BFGS",hessian=TRUE
)

bstar=opt_out$par  ## Posterior mode for adjusted model
bstar
bstar+as.vector(mu)  # mode for actual model
A1=opt_out$hessian # Approximate Precision at mode

## Standardize Model

Standard_Mod=glmb_Standardize_Model(y=as.vector(y), x=as.matrix(x),P=as.matrix(P),
                                    bstar=as.matrix(bstar,ncol=1), A1=as.matrix(A1))

bstar2=Standard_Mod$bstar2
A=Standard_Mod$A
x2=Standard_Mod$x2
mu2=Standard_Mod$mu2
P2=Standard_Mod$P2
L2Inv=Standard_Mod$L2Inv
L3Inv=Standard_Mod$L3Inv

Env2=EnvelopeBuild(as.vector(bstar2), as.matrix(A),y, as.matrix(x2),
as.matrix(mu2,ncol=1),as.matrix(P2),as.vector(alpha),as.vector(wt2),
family="binomial",link="logit",Gridtype=as.integer(3), n=as.integer(n),
sortgrid=TRUE)

## These now seem to match

Env2
```

`EnvelopeOpt` *Optimizes Envelope function for simulation*

Description

Optimizes the size of the grid to try to limit the combined time of the envelope construction and the simulation phase.

Usage

```
EnvelopeOpt(a1, n)
```

Arguments

`a1`	Diagonal elements of data precision matrix for a model in standard form
`n`	Number of draws to generate

Details

This function attempts to find a computationally optimal gridsize by using information on the strenght of the prior and the number of iterations desired. Generally, more data (i.e., larger values for the diagonal elements of the precision matrix) will require a larger grid. The same also holds when the number of desired draws is higher (as the setup costs associated with the larger grid is offset by the savings in the number of candidates per sample).

Value

A vector containing information on how many component each dimension should be split into.

See Also

rglmb, EnvelopeBuild, EnvelopeSort

Examples

```
data(menarche2)

summary(menarche2)
plot(Menarche/Total ~ Age, data=menarche2)

Age2=menarche2$Age-13

x<-matrix(as.numeric(1.0),nrow=length(Age2),ncol=2)
x[,2]=Age2

y=menarche2$Menarche/menarche2$Total
wt=menarche2$Total

mu<-matrix(as.numeric(0.0),nrow=2,ncol=1)
mu[2,1]=(log(0.9/0.1)-log(0.5/0.5))/3

V1<-1*diag(as.numeric(2.0))

# 2 standard deviations for prior estimate at age 13 between 0.1 and 0.9
## Specifies uncertainty around the point estimates

V1[1,1]<-((log(0.9/0.1)-log(0.5/0.5))/2)^2
V1[2,2]=(3*mu[2,1]/2)^2  # Allows slope to be up to 1 times as large as point estimate

famfunc<-glmbfamfunc(binomial(logit))

f1<-famfunc$f1
f2<-famfunc$f2
f3<-famfunc$f3
f5<-famfunc$f5
f6<-famfunc$f6

dispersion2<-as.numeric(1.0)
start <- mu
offset2=rep(as.numeric(0.0),length(y))
P=solve(V1)
n=1000
```

```
###### Adjust weight for dispersion

wt2=wt/dispersion2

######################### Shift mean vector to offset so that adjusted model has 0 mean

alpha=x%*%as.vector(mu)+offset2
mu2=0*as.vector(mu)
P2=P
x2=x


#####  Optimization step to find posterior mode and associated Precision

parin=start-mu

opt_out=optim(parin,f2,f3,y=as.vector(y),x=as.matrix(x),mu=as.vector(mu2),
              P=as.matrix(P),alpha=as.vector(alpha),wt=as.vector(wt2),
              method="BFGS",hessian=TRUE
)

bstar=opt_out$par  ## Posterior mode for adjusted model
bstar
bstar+as.vector(mu)  # mode for actual model
A1=opt_out$hessian # Approximate Precision at mode

## Standardize Model

Standard_Mod=glmb_Standardize_Model(y=as.vector(y), x=as.matrix(x),
P=as.matrix(P),bstar=as.matrix(bstar,ncol=1), A1=as.matrix(A1))

bstar2=Standard_Mod$bstar2
A=Standard_Mod$A
x2=Standard_Mod$x2
mu2=Standard_Mod$mu2
P2=Standard_Mod$P2
L2Inv=Standard_Mod$L2Inv
L3Inv=Standard_Mod$L3Inv

Env2=EnvelopeBuild(as.vector(bstar2), as.matrix(A),y, as.matrix(x2),
as.matrix(mu2,ncol=1),as.matrix(P2),as.vector(alpha),as.vector(wt2),
family="binomial",link="logit",Gridtype=as.integer(3),
n=as.integer(n),sortgrid=TRUE)

## These now seem to match

Env2
```

EnvelopeSort *Sorts Envelope function for simulation*

Description

Sorts Enveloping function for simulation. how frequently each component of the resulting grid should be sampled during simulation.

Usage

```
EnvelopeSort(
  l1,
  l2,
  GIndex,
  G3,
  cbars,
  logU,
  logrt,
  loglt,
  logP,
  LLconst,
  PLSD,
  a1
)
```

Arguments

l1	dimension for model (number of independent variables in X matrix)
l2	dimension for Envelope (number of components)
GIndex	matrix containing information on how each dimension should be sampled (1 means left tail of a restricted normal, 2 center, 3 right tail, and 4 the entire line)
G3	A matrix containing the points of tangencies associated with each component of the grid
cbars	A matrix containing the gradients for the negative log-likelihood at each tangency
logU	A matrix containing the log of the cummulative probability associated with each dimension
logrt	A matrix containing the log of the probability associated with the right tail (i.e. that to the right of the lower bound)
loglt	A matrix containing the log of the probability associated with the left tail (i.e., that to the left of the upper bound)
logP	A matrix containing log-probabilities related to the components of the grid
LLconst	A vector containing constant for each component of the grid used during the accept-reject procedure
PLSD	A vector containing the probability of each component in the Grid
a1	A vector containing the diagonal of the standardized precision matrix

Details

This function sorts the envelope in descending order based on the probability associated with each component in the Grid. Sorting helps speed up simulation once the envelope is constructed.

Value

The function returns a list consisting of the following components (the first six of which are matrics with number of rows equal to the number of components in the Grid and columns equal to the number of parameters):

GridIndex	A matrix containing information on how each dimension should be sampled (1 means left tail of a restricted normal, 2 center, 3 right tail, and 4 the entire line)
thetabars	A matrix containing the points of tangencies associated with each component of the grid
cbars	A matrix containing the gradients for the negative log-likelihood at each tangency
logU	A matrix containing the log of the cummulative probability associated with each dimension
logrt	A matrix containing the log of the probability associated with the right tail (i.e. that to the right of the lower bound)
loglt	A matrix containing the log of the probability associated with the left tail (i.e., that to the left of the upper bound)
LLconst	A vector containing constant for each component of the grid used during the accept-reject procedure
logP	A matrix containing log-probabilities related to the components of the grid
PLSD	A vector containing the probability of each component in the Grid

Examples

```
data(menarche2)

summary(menarche2)
plot(Menarche/Total ~ Age, data=menarche2)

Age2=menarche2$Age-13

x<-matrix(as.numeric(1.0),nrow=length(Age2),ncol=2)
x[,2]=Age2

y=menarche2$Menarche/menarche2$Total
wt=menarche2$Total

mu<-matrix(as.numeric(0.0),nrow=2,ncol=1)
mu[2,1]=(log(0.9/0.1)-log(0.5/0.5))/3

V1<-1*diag(as.numeric(2.0))

# 2 standard deviations for prior estimate at age 13 between 0.1 and 0.9
## Specifies uncertainty around the point estimates

V1[1,1]<-((log(0.9/0.1)-log(0.5/0.5))/2)^2
V1[2,2]=(3*mu[2,1]/2)^2  # Allows slope to be up to 1 times as large as point estimate

famfunc<-glmbfamfunc(binomial(logit))

f1<-famfunc$f1
f2<-famfunc$f2
f3<-famfunc$f3
f5<-famfunc$f5
f6<-famfunc$f6

dispersion2<-as.numeric(1.0)
start <- mu
```

```
offset2=rep(as.numeric(0.0),length(y))
P=solve(V1)
n=1000

## Appears that the type for some of these arguments are important/problematic

#outlist<-rnnorm_reg_cpp(n=as.integer(n),y=as.vector(y),x=as.matrix(x),
#mu=as.vector(mu),P=as.matrix(P),offset2=as.vector(offset2),
#wt=as.vector(wt),dispersion=as.numeric(dispersion2),famfunc=famfunc,
#f1=f1,f2=f2,f3=f3,start=as.vector(start),family="binomial",
#link="logit",Gridtype=as.integer(3))

### This allows use of the rglmb summary function
### add interface for glmbsim_NGauss_cpp later

#outlist$call<-match.call()
#colnames(outlist$coefficients)<-colnames(x)
#class(outlist)<-c(outlist$class,"rglmb")
#summary(outlist)
#Env1=outlist$Envelope


###### Adjust weight for dispersion

wt2=wt/dispersion2

######################### Shift mean vector to offset so that adjusted model has 0 mean

alpha=x%*%as.vector(mu)+offset2
mu2=0*as.vector(mu)
P2=P
x2=x


#####  Optimization step to find posterior mode and associated Precision

parin=start-mu

opt_out=optim(parin,f2,f3,y=as.vector(y),x=as.matrix(x),mu=as.vector(mu2),
              P=as.matrix(P),alpha=as.vector(alpha),wt=as.vector(wt2),
              method="BFGS",hessian=TRUE
)

bstar=opt_out$par  ## Posterior mode for adjusted model
bstar
bstar+as.vector(mu)  # mode for actual model
A1=opt_out$hessian # Approximate Precision at mode

## Standardize Model

Standard_Mod=glmb_Standardize_Model(y=as.vector(y), x=as.matrix(x),
P=as.matrix(P),bstar=as.matrix(bstar,ncol=1), A1=as.matrix(A1))

bstar2=Standard_Mod$bstar2
A=Standard_Mod$A
x2=Standard_Mod$x2
mu2=Standard_Mod$mu2
```

```
P2=Standard_Mod$P2
L2Inv=Standard_Mod$L2Inv
L3Inv=Standard_Mod$L3Inv

Env2=EnvelopeBuild(as.vector(bstar2), as.matrix(A),y, as.matrix(x2),
as.matrix(mu2,ncol=1),as.matrix(P2),as.vector(alpha),as.vector(wt2),
family="binomial",link="logit",Gridtype=as.integer(3),
n=as.integer(n),sortgrid=TRUE)

## These now seem to match

#Env1
Env2
```

`extractAIC.glmb` *Extract DIC from a Fitted Bayesian Model*

Description

Computes the Deviance Information Criterion for a fitted Bayesian Generalized Linear Model

Usage

```
## S3 method for class 'glmb'
extractAIC(fit, ...)

extractDIC(fit, ...)
```

Arguments

`fit`	an object of class `"glmb"`, typically the result of a call to glmb
`...`	further arguments to or from other methods

Value

A list with the Estimated effective number of parameters `pD` and the `DIC` from the object `fit` of class `"glmb"`.

Examples

```
## ----dobson----------------------------------------------------------------
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)

## Prior mean vector
mu<-matrix(0,5)
mu[1,1]=log(mean(counts))
## Prior standard deviation and Variance
mysd<-1
```

```
V=((mysd)^2)*diag(5)
## Call to glmb
glmb.D93<-glmb(n=1000,counts ~ outcome + treatment,
               family = poisson(),pfamily=dNormal(mu=mu,Sigma=V))
## ----glmb extractAIC-------------------------------------------------------
extractAIC(glmb.D93)
```

`extractAIC.rglmb` *Extract DIC from a Fitted Bayesian Model*

Description

Computes the Deviance Information Criterion for a fitted Bayesian Generalized Linear Model

Usage

```
## S3 method for class 'rglmb'
extractAIC(fit, ...)
```

Arguments

`fit`	an object of class `"rglmb"`, typically the result of a call to rglmb
`...`	further arguments to or from other methods

Value

A list with the Estimated effective number of parameters `pD` and the `DIC` from the object `fit` of class `"glmb"`.

Examples

```
## ----dobson----------------------------------------------------------------
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)

## Prior mean vector
mu<-matrix(0,5)
mu[1,1]=log(mean(counts))
## Prior standard deviation and Variance
mysd<-1
V=((mysd)^2)*diag(5)
## Call to glmb
glmb.D93<-glmb(n=1000,counts ~ outcome + treatment,
               family = poisson(),pfamily=dNormal(mu=mu,Sigma=V))
## ----glmb extractAIC-------------------------------------------------------
extractAIC(glmb.D93)
```

`f2_gaussian_vector` *Derives the inverse of the gradient vector.*

Description

Computes the inverse of the gradient vector for the gaussian model. Typically done to find the set of tangency points that yield the same gradient as an initial set of gradients used for an envelope positioned at the posterior mode for a specific dispersion.

Usage

```
f2_gaussian_vector(b, y, x, mu, P, alpha, wt)
```

Arguments

`b`	A matrix storing a set of vectors for which the negative log-likelihood is to be evaluated.
`x, y`	For `glm`: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For `glm.fit`: `x` is a design matrix of dimension `n * p`, and `y` is a vector of observations of length `n`.
`mu`	Prior mean
`P`	Prior Precision matrix
`alpha`	offset vector
`wt`	weighting vector

Value

Refer to Nygren and Nygren (2006) for details. The first set of items refers to Example 2 in section 3.1. All except the last item in this list of returned items has a number of rows equaling the number of components of the grid and a number of columns equaling the number of coefficients in the model. All quantities refer to the respective coefficient for each of the components of the grid.

`Down`	The lower bounds for the interval to be evaluated. Either negative infinity or a real number.
`Up`	The upper bounds for the interval to be evaluated. Either positive infinity or a real number.
`lglt`	The log of the density between negative infinity and the upper bound
`lgrt`	The log of the density between the lower bound and infinity
`lgct`	The log of the density between the lower and upper bounds
`logU`	The one of the 3 above that is relevant for the component of the grid
`logP`	A two column matrix, the first of which holds sum of logU across the components. The second column is 0 and is later populated by the Set_logP function

Examples

```
## ----dobson-----------------------------------------------------------------
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)

## Prior mean vector
mu<-matrix(0,5)
mu[1,1]=log(mean(counts))
## Prior standard deviation and Variance
mysd<-1
V=((mysd)^2)*diag(5)
## Call to glmb
glmb.D93<-glmb(n=1000,counts ~ outcome + treatment,
               family = poisson(),pfamily=dNormal(mu=mu,Sigma=V))
## ----glmb extractAIC--------------------------------------------------------
extractAIC(glmb.D93)
```

---

formula.summary.rglmb *Extract Log-Likelihood*

Description

This function is a method function for the "glmb" class used to Extract the log-Likelihood from a Bayesian Generalized Linear Model.

Usage

```
## S3 method for class 'summary.rglmb'
formula(x, ...)
```

Arguments

x	an object of class glmb, typically the result of a call to glmb
...	further arguments to or from other methods

Value

The function returns a vector, logLikout with the estimated log-likelihood for each draw.

Examples

```
## ----dobson-----------------------------------------------------------------
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)

## Prior mean vector
mu<-matrix(0,5)
mu[1,1]=log(mean(counts))
## Prior standard deviation and Variance
```

```
mysd<-1
V=((mysd)^2)*diag(5)
## Call to glmb
glmb.D93<-glmb(n=1000,counts ~ outcome + treatment,
               family = poisson(),pfamily=dNormal(mu=mu,Sigma=V))
## ----glmb logLik-----------------------------------------------------------
colMeans(logLik(glmb.D93))
```

---

glmb *Fitting Bayesian Generalized Linear Models*

Description

glmb is used to fit Bayesian generalized linear models, specified by giving a symbolic descriptions of the linear predictor, the error distribution, and the prior distribution.

Usage

```
glmb(
  formula,
  family = binomial,
  pfamily = dNormal(mu, Sigma, dispersion = 1),
  n = 1000,
  data,
  weights,
  subset,
  offset,
  na.action,
  Gridtype = 1,
  start = NULL,
  etastart,
  mustart,
  control = list(...),
  model = TRUE,
  method = "glm.fit",
  x = FALSE,
  y = TRUE,
  contrasts = NULL,
  ...
)

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

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.

family	a description of the error distribution and link function to be used in the model. For glm this can be a character string naming a family function, a family function or the result of a call to a family function. For glm.fit only the third option is supported. (See family for details of family functions.)
pfamily	a description of the prior distribution and associated constants to be used in the model. This should be a pfamily function (see pfamily for details of pfamily functions).
n	number of draws to generate. If length(n) > 1, the length is taken to be the number required.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which glm is called.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
offset	this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See documentation for model.offset at model.extract.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The 'factory-fresh' default is stats{na.omit}. Another possible value is NULL, no action. Value stats{na.exclude} can be useful.
Gridtype	an optional argument specifying the method used to determine the number of tangent points used to construct the enveloping function.
start	starting values for the parameters in the linear predictor.
etastart	starting values for the linear predictor.
mustart	starting values for the vector of means.
control	a list of parameters for controlling the fitting process. For glm.fit this is passed to glm.control.
model	a logical value indicating whether *model frame* should be included as a component of the returned value.
method	the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS): the alternative "model.frame" returns the model frame and does no fitting. User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If specified as a character string it is looked up from within the **stats** namespace.
x, y	For glm: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For glm.fit: x is a design matrix of dimension n * p, and y is a vector of observations of length n.

contrasts	an optional list. See the contrasts.arg of model.matrix.default.
...	For glm: arguments to be used to form the default control argument if it is not supplied directly. For weights: further arguments passed to or from other methods.
digits	the number of significant digits to use when printing.

Details

The function glmb is a Bayesian version of the classical glm function. Setup of the models mirrors those for the glm function but add additional required arguments mu and Sigma, representing a multivariate normal prior. In addition, the dispersion parameter must currently be provided for the gaussian, Gamma, quasipoisson, and quasibinomial families (future implementations may incoporate priors for these into the glmb function). The function generates random iid samples from the posterior density associated with the model (assuming a fixed dispersion parameter).

Current implemented models include the gaussian family (identity link function), the poisson and quasipoisson families (log link function), the gamma family (log link function), as well as the binomial and quasibinomial families (logit, probit, and cloglog link functions).

The function returns the output from a call to the function glm as well as the simulated Bayesian coefficients and associated outputs. In addition, the function returns model diagnostic information related to the DIC, a Bayesian Information Criterion similar to the AIC for classical models.

For the gaussian family, iid samples from the posterior density are genererated using standard simulation procedures for multivariate normal densities. For all other families, the samples are generated using accept-reject procedures by leveraging the likelihood subgradient approach of Nygren and Nygren (2006). This approach relies on tight enveloping functions that bound the posterior density from above. The Gridtype parameter is used to select the method used for determining the size of a Grid used to build the enveloping function. See EnvelopeBuild for details. Depending on the selection, the time to build the envelope and the acceptance rate during the simulation process may vary. The returned value iters contains the number of candidates generated before acceptance for each draw.

Value

glmb returns an object of class "glmb". The function summary (i.e., summary.glmb) can be used to obtain or print a summary of the results. The generic accessor functions coefficients, fitted.values, residuals, and extractAIC can be used to extract various useful features of the value returned by glmb.

An object of class "glmb" is a list containing at least the following components:

glm	an object of class "glm" containing the output from a call to the function glm
coefficients	a matrix of dimension n by length(mu) with one sample in each row
coef.means	a vector of length(mu) with the estimated posterior mean coefficients
coef.mode	a vector of length(mu) with the estimated posterior mode coefficients
dispersion	Either a constant provided as part of the call, or a vector of length n with one sample in each row.
Prior	A list with the priors specified for the model in question. Items in list may vary based on the type of prior
fitted.values	a matrix of dimension n by length(y) with one sample for the mean fitted values in each row
family	the family object used.

linear.predictors	an n by length(y) matrix with one sample for the linear fit on the link scale in each row
deviance	an n by 1 matrix with one sample for the deviance in each row
pD	An Estimate for the effective number of parameters
Dbar	Expected value for minus twice the log-likelihood function
Dthetabar	Value of minus twice the log-likelihood function evaluated at the mean value for the coefficients
DIC	Estimated Deviance Information criterion
prior.weights	a vector of weights specified or implied by the model
y	a vector with the dependent variable
x	a matrix with the implied design matrix for the model
model	if requested (the default),the model frame
call	the matched call
formula	the formula supplie
terms	the terms object used
data	the data argument
famfunc	Family functions used during estimation process
iters	an n by 1 matrix giving the number of candidates generated before acceptance for each sample.
contrasts	(where relevant) the contrasts used.
xlevels	(where relevant) a record of the levels of the factors used in fitting
digits	the number of significant digits to use when printing.

In addition, non-empty fits will have (yet to be implemented) components qr, R and effects relating to the final weighted linear fit for the posterior mode. Objects of class "glmb" are normall of class c("glmb","glm","lm"), that is inherit from classes glm and lm and well-designed methods from those classed will be applied when appropriate.

If a binomial glmb model was specified by giving a two-column response, the weights returned by prior.weights are the total number of cases (factored by the supplied case weights) and the component of y of the result is the proportion of successes.

Author(s)

The R implementation of glmb has been written by Kjell Nygren and was built to be a Bayesian version of the glm function and hence tries to mirror the features of the glm function to the greatest extent possible. For details on the author(s) for the glm function see the documentation for glm.

References

Dobson, A. J. (1990) *An Introduction to Generalized Linear Models.* London: Chapman and Hall.

Hastie, T. J. and Pregibon, D. (1992) *Generalized linear models.* Chapter 6 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole. McCullagh P. and Nelder, J. A. (1989) *Generalized Linear Models.* London: Chapman and Hall.

Nygren, K.N. and Nygren, L.M (2006) Likelihood Subgradient Densities. *Journal of the American Statistical Association*. vol.101, no.475, pp 1144-1156. doi: 10.1198/016214506000000357.

Raiffa, Howard and Schlaifer, R (1961) *Applied Statistical Decision Theory.* Boston: Clinton Press, Inc.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* New York: Springer.

See Also

lm and glm for classical modeling functions.

pfamily for documentation of pfamily functions used to specify priors.

Prior_Setup, Prior_Check for functions used to initialize and to check priors,

summary.glmb, predict.glmb, residuals.glmb, simulate.glmb, extractAIC.glmb, dummy.coef.glmb and methods(class="glmb") for glmb and the methods and generic functions for classes glm and lm from which class glmb inherits.

Other glmbayes modeling functions: lmb(), rglmb(), rlmb()

Examples

```
set.seed(333)
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
mysd<-1
mu<-matrix(0,5)
mu[1]=log(mean(counts))
V0<-((mysd)^2)*diag(5)
glmb.D93<-glmb(counts ~ outcome + treatment, family = poisson(),pfamily=dNormal(mu=mu,Sigma=V0))
summary(glmb.D93)

# Menarche Binomial Data Example
data(menarche2)
Age2=menarche2$Age-13

#Priors Are Derived in Vignette
nvars=2
## Logit Prior and Model
mu1=as.matrix(c(0,1.098612),ncol=1)
V1<-1*diag(nvars)
V1[1,1]=0.18687882
V1[2,2]=0.10576217
V1[1,2]=-0.03389182
V1[2,1]=-0.03389182
glmb.out1<-glmb(cbind(Menarche, Total-Menarche) ~ Age2,
family=binomial(logit),pfamily=dNormal(mu=mu1,Sigma=V1), data=menarche2)
summary(glmb.out1)

## Probit Prior and Model
mu2=as.matrix(c(0,0.6407758),ncol=1)
V2<-1*diag(nvars)
V2[1,1]=0.07158369
V2[2,2]=0.03453205
V2[1,2]=-0.01512075
V2[2,1]=-0.01512075
glmb.out2<-glmb(cbind(Menarche, Total-Menarche) ~ Age2,
family=binomial(probit),pfamily=dNormal(mu=mu2,Sigma=V2), data=menarche2)
summary(glmb.out2)

## clog-log Prior and Model
mu2=as.matrix(c(-0.3665129,0.6002727),ncol=1)
V2<-1*diag(nvars)
V2[1,1]=0.11491322
```

```
V2[2,2]=0.03365986
V2[1,2]=-0.03502783
V2[2,1]=-0.03502783
glmb.out3<-glmb(cbind(Menarche, Total-Menarche) ~ Age2,
              family=binomial(cloglog),pfamily=dNormal(mu=mu2,Sigma=V2), data=menarche2)
summary(glmb.out3)

DIC_Out=rbind(extractAIC(glmb.out1),extractAIC(glmb.out2),extractAIC(glmb.out3))
rownames(DIC_Out)=c("logit","probit","clog-log")
colnames(DIC_Out)=c("pD","DIC")
DIC_Out
```

`glmb.influence.measures` *Bayesian Regression Diagnostics*

Description

This function provides the basic quantities which are used in forming a wide variety of diagnostics for checking the quality of Bayesian regression fits.

Usage

```
glmb.influence.measures(model, infl = influence(model))

## S3 method for class 'glmb'
rstandard(model, ..., infl = influence(model))

## S3 method for class 'glmb'
rstudent(model, ..., infl = influence(model))

glmb.dffits(model, infl = influence(model))

## S3 method for class 'glmb'
dfbetas(model, ..., infl = influence(model))

glmb.covratio(model, infl = influence(model))

## S3 method for class 'glmb'
cooks.distance(model, ..., infl = influence(model))
```

Arguments

`model`	an R object, typically returned by `lm` or `glm`.
`infl`	influence structure as returned by `influence.glmb` (the latter only for the glm method of `rstudent` and `cooks.distance`).
`...`	further arguments passed to or from other methods.

Value

a `list` wih components:

Examples

```
set.seed(333)
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
mysd<-1
mu<-matrix(0,5)
mu[1]=log(mean(counts))
V0<-((mysd)^2)*diag(5)
glmb.D93<-glmb(counts ~ outcome + treatment, family = poisson(),pfamily=dNormal(mu=mu,Sigma=V0))

## Start setup here [First output from earlier optim optimization]
betastar=glmb.D93$coef.mode  # Posterior mode from optim
x=glmb.D93$x
y=glmb.D93$y
#offset=glmb.D93$offset   # not present in the output --> For now set to 0 vector
offset2=0*y   # Should return this from lower level functions
weights2=glmb.D93$prior.weights

## Check influence measures for original model
fit=glmb.wfit(x,y,weights2,offset2,family=poisson(),Bbar=mu,P=solve(V0),betastar)
influence.measures(fit)

print(fit)
print(glmb.D93$coef.mode)

### Now try a strong prior with poorly chosen intercept
mu1=0*mu
V1=0.1*V0
glmb2.D93<-glmb(counts ~ outcome + treatment, family = poisson(),pfamily=dNormal(mu=mu1,Sigma=V1))

Bbar2=mu1  # Prior mean
betastar2=glmb2.D93$coef.mode  # Posterior mode from optim
fit2=glmb.wfit(x,y,weights2,offset2,family=poisson(),Bbar2,P=solve(V1),betastar2)

influence.measures(fit2)

print(fit2)
print(glmb2.D93$coef.mode)



influence(glmb2.D93)
influence(glmb2.D93,do.coef=TRUE)
influence(glmb2.D93,do.coef=FALSE)

#glmb2.D93$qr=glmb2.D93$fit$qr
#influence.measures(glmb2.D93,influence(glmb2.D93))
#influence.measures(glmb2.D93$fit,influence(glmb2.D93))
#influence.measures(glmb2.D93$fit,influence(glmb2.D93,do.coef=FALSE))
glmb.influence.measures(glmb2.D93,influence(glmb2.D93))


## Do not need method functions
dfbeta(glmb2.D93,influence(glmb2.D93))
```

```
dfbeta(glmb2.D93$fit,influence(glmb2.D93))
hatvalues(glmb2.D93,influence(glmb2.D93))
hatvalues(glmb2.D93$fit,influence(glmb2.D93))

# Methods (implmented)

rstandard(glmb2.D93,infl=influence(glmb2.D93))
rstandard(glmb2.D93)


# Need Metods

## Needs a method function
#dfbetas(glmb2.D93,influence(glmb2.D93))
dfbetas(glmb2.D93)
dfbetas(glmb2.D93,influence(glmb2.D93,do.coef=TRUE))
dfbetas(glmb2.D93$fit,influence(glmb2.D93))




# Needs a method function
cooks.distance(glmb2.D93)
cooks.distance(glmb2.D93$fit,influence(glmb2.D93))



# Needs a method function (now works)
#rstandard(glmb2.D93$fit,influence(glmb2.D93))


# Needs a method function
rstudent(glmb2.D93)
rstudent(glmb2.D93$fit,influence(glmb2.D93))


# Not a method, separate function
glmb.dffits(glmb2.D93)
dffits(glmb2.D93$fit,influence(glmb2.D93))

# Not a method - separate function
glmb.covratio(glmb2.D93)
covratio(glmb2.D93$fit,influence(glmb2.D93))



# Needs a method function but requires a different approach
# The influence function actually stores this measure
## This seems to require more work to create a method function
## Should
#hat(glmb2.D93,influence(glmb2.D93))
#hat(glmb2.D93$fit,influence(glmb2.D93))

infl=influence(glmb2.D93)
hat2=infl$hat
hat2
```

glmb.wfit *Fitter Function for Bayesian Generalized Linear Models*

Description

Basic computing engine called to replicate the output from the lm.fit function for an already optimized Bayesian Generalized Linear model.

Usage

```
glmb.wfit(
  x,
  y,
  weights = rep.int(1, nobs),
  offset = rep.int(0, nobs),
  family = gaussian(),
  Bbar,
  P,
  betastar,
  method = "qr",
  tol = 1e-07,
  singular.ok = TRUE,
  ...
)
```

Arguments

x	design matrix of dimension n * p.
y	vector of observations of length n, or a matrix with n rows.
weights	an optional vector of *prior weights* to be used in the fitting process. Should be NULL or a numeric vector.
offset	(numeric of length n). This can be used to specify an *a priori* known component to be included in the linear predictor during fitting.
family	a description of the error distribution and link function to be used in the model. Should be a family function. (see family for details of family functions.)
Bbar	Prior mean vector of length p.
P	Prior precision matrix of dimension p * p.
betastar	Posterior mode vector of length p which has already been estimated.
method	currently, only method = "qr" is supported.
tol	tolerance for the qr decomposition. Default is 1e-7.
singular.ok	logical. If FALSE, a singular model is an error.
...	currently disregarded.

Value

a list wih components:

Examples

```
set.seed(333)
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
mysd<-1
mu<-matrix(0,5)
mu[1]=log(mean(counts))
V0<-((mysd)^2)*diag(5)
glmb.D93<-glmb(counts ~ outcome + treatment, family = poisson(),pfamily=dNormal(mu=mu,Sigma=V0))

## Start setup here [First output from earlier optim optimization]
betastar=glmb.D93$coef.mode  # Posterior mode from optim
x=glmb.D93$x
y=glmb.D93$y
#offset=glmb.D93$offset   # not present in the output --> For now set to 0 vector
offset2=0*y   # Should return this from lower level functions
weights2=glmb.D93$prior.weights

## Check influence measures for original model
fit=glmb.wfit(x,y,weights2,offset2,family=poisson(),Bbar=mu,P=solve(V0),betastar)
influence.measures(fit)

print(fit)
print(glmb.D93$coef.mode)

### Now try a strong prior with poorly chosen intercept
mu1=0*mu
V1=0.1*V0
glmb2.D93<-glmb(counts ~ outcome + treatment, family = poisson(),pfamily=dNormal(mu=mu1,Sigma=V1))

Bbar2=mu1  # Prior mean
betastar2=glmb2.D93$coef.mode  # Posterior mode from optim
fit2=glmb.wfit(x,y,weights2,offset2,family=poisson(),Bbar2,P=solve(V1),betastar2)

influence.measures(fit2)

print(fit2)
print(glmb2.D93$coef.mode)



influence(glmb2.D93)
influence(glmb2.D93,do.coef=TRUE)
influence(glmb2.D93,do.coef=FALSE)

#glmb2.D93$qr=glmb2.D93$fit$qr
#influence.measures(glmb2.D93,influence(glmb2.D93))
#influence.measures(glmb2.D93$fit,influence(glmb2.D93))
#influence.measures(glmb2.D93$fit,influence(glmb2.D93,do.coef=FALSE))
glmb.influence.measures(glmb2.D93,influence(glmb2.D93))


## Do not need method functions
dfbeta(glmb2.D93,influence(glmb2.D93))
```

```
dfbeta(glmb2.D93$fit,influence(glmb2.D93))
hatvalues(glmb2.D93,influence(glmb2.D93))
hatvalues(glmb2.D93$fit,influence(glmb2.D93))

# Methods (implmented)

rstandard(glmb2.D93,infl=influence(glmb2.D93))
rstandard(glmb2.D93)


# Need Metods

## Needs a method function
#dfbetas(glmb2.D93,influence(glmb2.D93))
dfbetas(glmb2.D93)
dfbetas(glmb2.D93,influence(glmb2.D93,do.coef=TRUE))
dfbetas(glmb2.D93$fit,influence(glmb2.D93))




# Needs a method function
cooks.distance(glmb2.D93)
cooks.distance(glmb2.D93$fit,influence(glmb2.D93))



# Needs a method function (now works)
#rstandard(glmb2.D93$fit,influence(glmb2.D93))


# Needs a method function
rstudent(glmb2.D93)
rstudent(glmb2.D93$fit,influence(glmb2.D93))


# Not a method, separate function
glmb.dffits(glmb2.D93)
dffits(glmb2.D93$fit,influence(glmb2.D93))

# Not a method - separate function
glmb.covratio(glmb2.D93)
covratio(glmb2.D93$fit,influence(glmb2.D93))



# Needs a method function but requires a different approach
# The influence function actually stores this measure
## This seems to require more work to create a method function
## Should
#hat(glmb2.D93,influence(glmb2.D93))
#hat(glmb2.D93$fit,influence(glmb2.D93))

infl=influence(glmb2.D93)
hat2=infl$hat
hat2
```

`glmbayes` *Bayesian Generalized Linear Models (iid Samples)*

Description

Generates iid samples for Bayesian Generalized Linear Models.

Usage

```
PackageOverview
```

Format

An object of class `NULL` of length 0.

Details

The `glmbayes` package produces iid samples for Bayesian Genereralized Linear Models and is intended as a Bayesian version of the `glm` function for classical models. Estimation can be performed using three main functions. For models with fixed dispersion parameters, the `rglmb` function is the workhorse function and comes with a minimialistic interface for the input and output. It is also suitable for use as part of block Gibbs sampling procedures. The `glmb` function is essentially a wrapper function for the rglmb function that provides an interface closer to that of the `glm` function. The `rGamma_reg` function can be leveraged in order to produce samples for the dispersion parameters associated with the gaussian and Gamma link functions. Most methods defined for the output of the `glm` function are also defined for the `glmb`, `rglmb`, and `rGamma_reg` functions (see their respective documentation for details).

For the regression parameters, multivariate normal priors are assumed. Simulation for the gaussian family with the identify link function is performed using standard procedures for multivariate normal densities. For all other families and link functions, simulation is performed using the likelihood subgradient approach of Nygren and Nygren (2006). This approach involves the construction of an enveloping function for the full posterior density followed by accept-reject sampling. For models that are approximately multivariate normal, the expected number of draws required per acceptance are bounded from above as noted in Nygren and Nygren (2006).

Currently implemented models include the gaussian (identity link), poisson/quasipoisson (log link), binomial/quasibinomial (logit, probit, and cloglog links), and Gamma (log link) families. These models all have log-concave likelihood functions that allow us to leverage the likelihood-subgradient approach for the iid sampling. Models that fail to have log-concave likelihood functions are not implemented. Our demos (viewable by entering the `demo()` command) provides examples of each of these families and links.

The current implementation requires separate use of the `rGamma_reg` function in order to generate samples for dispersion parameters (gaussian, Gammma, quasipoisson, quasi-binomial families). Our demos include examples of the joint use of the `rglmb` and `rGamma_reg` to produce samples for both regression and dispersion parameters using two-block Gibbs samplers. As these two-block Gibbs samplers likely are geometrically ergodic, future implementations may incorporate these two-block Gibbs samplers into the `rglmb` and `glmb` functions by leveraging theoretical bounds om convergence rates derived using Rosenthal (1996) type drift and minorization conditions.

The `rglmb` function can also be used in Block-Gibbs sampling implementations for Hierarchical Bayesian models. The demos associated with this package contains examples of such models.

Author(s)

Kjell Nygren

References

Dobson, A. J. (1990) *An Introduction to Generalized Linear Models.* London: Chapman and Hall.

Hastie, T. J. and Pregibon, D. (1992) *Generalized linear models.* Chapter 6 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole. McCullagh P. and Nelder, J. A. (1989) *Generalized Linear Models.* London: Chapman and Hall.

Nygren, K.N. and Nygren, L.M (2006) Likelihood Subgradient Densities. *Journal of the American Statistical Association*. vol.101, no.475, pp 1144-1156.

Raiffa, Howard and Schlaifer, R (1961) *Applied Statistical Decision Theory.* Boston: Clinton Press, Inc.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* New York: Springer.

Examples

```
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
print(d.AD <- data.frame(treatment, outcome, counts))

## Call to glm
glm.D93 <- glm(counts ~ outcome + treatment,
               family = poisson())

## Using glmb
## Step 1: Set up Prior
ps=Prior_Setup(counts ~ outcome + treatment)
mu=ps$mu
V=ps$Sigma
# Step2A: Check the Prior
Prior_Check(counts ~ outcome + treatment,family = poisson(),
            pfamily=dNormal(mu=mu,Sigma=V))
# Step2B: Update and Re-Check the Prior
mu[1,1]=log(mean(counts))
Prior_Check(counts ~ outcome + treatment,family = poisson(),
            pfamily=dNormal(mu=mu,Sigma=V))
# Step 3: Call the glmb function
glmb.D93<-glmb(counts ~ outcome + treatment, family=poisson(),
               pfamily=dNormal(mu=mu,Sigma=V))

## ----Printed_Views-------------------------------------------------------
## Printed view of the output from the glm function
print(glm.D93)
## Printed view of the output from the glmb function
print(glmb.D93)

## ----Methods-------------------------------------------------------------
## Methods for class "lm"
methods(class="lm")

## Methods for class "glm"
```

```
methods(class="glm")

## Methods for class "glmb"
methods(class="glmb")

## ----summary------------------------------------------------------------
## summary output for the "glm" class
summary(glm.D93)

## summary output for the "glm" class
summary(glmb.D93)

## ----fitted outputs-----------------------------------------------------
## fitted outputs for the glm function
fitted(glm.D93)

## ----glmb fitted outputs------------------------------------------------
## mean of fitted outputs for the glm function
colMeans(fitted(glmb.D93))

## ----predictions--------------------------------------------------------
## predictions for the glm function
predict(glm.D93)

## predictions for the glmb function
colMeans(predict(glmb.D93))

## ----residuals----------------------------------------------------------
## residuals for the glm function
residuals(glm.D93)

## residuals for the glmb function
colMeans(residuals(glmb.D93))

## ----vcov---------------------------------------------------------------
## vcov for the glm function
vcov(glm.D93)

## vcov for the glm function
vcov(glmb.D93)

## ----confint------------------------------------------------------------
## confint for the glm function
confint(glm.D93)

## confint for the glm function
confint(glmb.D93)

## ----AIC/DIC------------------------------------------------------------
## AIC for the glm function (equivalent degrees of freedom and the AIC)
extractAIC(glm.D93)

## DIC for the glmb function (estimated effective number of parameters and the DIC)
extractAIC(glmb.D93)

## ----Deviance-----------------------------------------------------------
## Deviance for the glm function
```

```
deviance(glm.D93)

## Deviance for the glmb function
mean(deviance(glmb.D93))

## ----logLik---------------------------------------------------------------
## Deviance for the glm function
logLik(glm.D93)

## Deviance for the glmb function
mean(logLik(glmb.D93))

## ----Model Frame----------------------------------------------------------
## Model Frame for the glm function
model.frame(glm.D93)

## Model Frame for the glmb function
model.frame(glmb.D93$glm)

## ----formula--------------------------------------------------------------
## formula for the glm function
formula(glm.D93)

## ----formula--------------------------------------------------------------
## formula for the glmb function
formula(glmb.D93)

## ----family---------------------------------------------------------------
## family for the glm function
family(glm.D93)

## family for the glmb function
family(glmb.D93$glm)

## ----nobs-----------------------------------------------------------------
## nobs for the glm function
nobs(glm.D93)

## nobs for the glmb function
nobs(glmb.D93)

## ----show-----------------------------------------------------------------
## show for the glm function
show(glm.D93)

## show for the glmb function
show(glmb.D93)
```

`glmbfamfunc` *Return family functions used during simulation and post processing*

Description

This function takes as input a `family` object and returns a set of functions that are used during simulation and summarization of models using the `glmb`, and `rglmb` functions.

Usage

```
glmbfamfunc(family)

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

Arguments

family	an object of class family
x	an object of class "glmbfamfunc" for which a printed output is desired.
...	additional optional arguments

Details

This function takes as input a family and returns a set of functions related to the family.

Value

A list with the following components

f1	Negative log-likelihood function
f2	Negative log-posterior function
f3	Gradient function for negative log-posterior function
f4	Deviance function
f7	Another function

Examples

```
famfunc<-glmbfamfunc(binomial(logit))

print(famfunc)

f1<-famfunc$f1
f2<-famfunc$f2
f3<-famfunc$f3
f5<-famfunc$f5
f6<-famfunc$f6
```

---

glmb_Standardize_Model

Standardize A Non-Gaussian Model

Description

Standardizes a Non-Gaussian Model prior to Envelope Creation

Usage

```
glmb_Standardize_Model(y, x, P, bstar, A1)
```

Arguments

y	a vector of observations of length m
x	a design matrix of dimension m*p
P	a positive-definite symmetric matrix specifying the prior precision matrix of the variables.
bstar	a matrix containing the posterior mode from an optimization step
A1	a matrix containing the posterior precision matrix at the posterior mode

Details

This functions starts with basic information about the model in the argument list and then uses the following steps to further standardize the model (the model is already assumed to have a 0 prior mean vector when this step is applied).

1) An eigenvalue composition is applied to the posterior precision matrix, and the model is (as an interim step) standardized to have a posterior precision matrix equal to the identity matrix. Please note that this means that the prior precision matrix after this step is "smaller" than the identity matrix.

2) A diagonal matrix epsilon is pulled out from the standardized prior precision matrix so that the remaining part of the prior precision matrix still is positive definite. That part is then treated as part of the posterior for the rest of the standardization and simulation and only the part connected to epsilon is treated as part of the prior. Note that the exact epsilon chosen seems not to matter. Hence there are many possible ways of doing this standardization and future versions of this package may tweak the current approach if it helps improve numerical accuracy or acceptance rates.

3) The model is next standardized (using a second eigenvalue decomposition) so that the prior (i.e., the portion connected to epsilon) is the identity matrix. The standardized model then simutaneously has the feature that the prior precision matrix is the identity matrix and that the data precision A (at the posterior mode) is a diagonal matrix. Hence the variables in the standardized model are approximately independent at the posterior mode.

Value

A list with the following components

bstar2	Standardized Posterior Mode
A	Standardized Data Precision Matrix
x2	Standardized Design Matrix
mu2	Standardized Prior Mean vector
P2	Standardized Precision Matrix Added to log-likelihood
L2Inv	A matrix used when undoing the first step in standardization described below
L3Inv	A matrix used when undoing the second step in standardization described below

Examples

```
data(menarche2)

summary(menarche2)
plot(Menarche/Total ~ Age, data=menarche2)

Age2=menarche2$Age-13
```

```
x<-matrix(as.numeric(1.0),nrow=length(Age2),ncol=2)
x[,2]=Age2

y=menarche2$Menarche/menarche2$Total
wt=menarche2$Total

mu<-matrix(as.numeric(0.0),nrow=2,ncol=1)
mu[2,1]=(log(0.9/0.1)-log(0.5/0.5))/3

V1<-1*diag(as.numeric(2.0))

# 2 standard deviations for prior estimate at age 13 between 0.1 and 0.9
## Specifies uncertainty around the point estimates

V1[1,1]<-((log(0.9/0.1)-log(0.5/0.5))/2)^2
V1[2,2]=(3*mu[2,1]/2)^2  # Allows slope to be up to 1 times as large as point estimate

famfunc<-glmbfamfunc(binomial(logit))

f1<-famfunc$f1
f2<-famfunc$f2
f3<-famfunc$f3
f5<-famfunc$f5
f6<-famfunc$f6

dispersion2<-as.numeric(1.0)
start <- mu
offset2=rep(as.numeric(0.0),length(y))
P=solve(V1)
n=1000

## Appears that the type for some of these arguments are important/problematic



###### Adjust weight for dispersion

wt2=wt/dispersion2

######################### Shift mean vector to offset so that adjusted model has 0 mean

alpha=x%*%as.vector(mu)+offset2
mu2=0*as.vector(mu)
P2=P
x2=x


#####  Optimization step to find posterior mode and associated Precision

parin=start-mu

opt_out=optim(parin,f2,f3,y=as.vector(y),x=as.matrix(x),mu=as.vector(mu2),
              P=as.matrix(P),alpha=as.vector(alpha),wt=as.vector(wt2),
              method="BFGS",hessian=TRUE
)
```

```
bstar=opt_out$par  ## Posterior mode for adjusted model
bstar
bstar+as.vector(mu)  # mode for actual model
A1=opt_out$hessian # Approximate Precision at mode

## Standardize Model

Standard_Mod=glmb_Standardize_Model(y=as.vector(y), x=as.matrix(x),P=as.matrix(P),
                                    bstar=as.matrix(bstar,ncol=1), A1=as.matrix(A1))

bstar2=Standard_Mod$bstar2
A=Standard_Mod$A
x2=Standard_Mod$x2
mu2=Standard_Mod$mu2
P2=Standard_Mod$P2
L2Inv=Standard_Mod$L2Inv
L3Inv=Standard_Mod$L3Inv
```

`influence.glmb` *Bayesian Regression Diagnostics*

Description

This function provides the basic quantities which are used in forming a wide variety of diagnostics for checking the quality of Bayesian regression fits.

Usage

```
## S3 method for class 'glmb'
influence(model, ...)
```

Arguments

`model`	an object as returned by `lm` or `glm`.
`...`	further arguments passed to or from other methods.

Value

a `list` wih components:

Examples

```
set.seed(333)
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
mysd<-1
mu<-matrix(0,5)
mu[1]=log(mean(counts))
V0<-((mysd)^2)*diag(5)
glmb.D93<-glmb(counts ~ outcome + treatment, family = poisson(),pfamily=dNormal(mu=mu,Sigma=V0))

## Start setup here [First output from earlier optim optimization]
```

```
betastar=glmb.D93$coef.mode  # Posterior mode from optim
x=glmb.D93$x
y=glmb.D93$y
#offset=glmb.D93$offset   # not present in the output --> For now set to 0 vector
offset2=0*y   # Should return this from lower level functions
weights2=glmb.D93$prior.weights

## Check influence measures for original model
fit=glmb.wfit(x,y,weights2,offset2,family=poisson(),Bbar=mu,P=solve(V0),betastar)
influence.measures(fit)

print(fit)
print(glmb.D93$coef.mode)

### Now try a strong prior with poorly chosen intercept
mu1=0*mu
V1=0.1*V0
glmb2.D93<-glmb(counts ~ outcome + treatment, family = poisson(),pfamily=dNormal(mu=mu1,Sigma=V1))

Bbar2=mu1  # Prior mean
betastar2=glmb2.D93$coef.mode  # Posterior mode from optim
fit2=glmb.wfit(x,y,weights2,offset2,family=poisson(),Bbar2,P=solve(V1),betastar2)

influence.measures(fit2)

print(fit2)
print(glmb2.D93$coef.mode)



influence(glmb2.D93)
influence(glmb2.D93,do.coef=TRUE)
influence(glmb2.D93,do.coef=FALSE)

#glmb2.D93$qr=glmb2.D93$fit$qr
#influence.measures(glmb2.D93,influence(glmb2.D93))
#influence.measures(glmb2.D93$fit,influence(glmb2.D93))
#influence.measures(glmb2.D93$fit,influence(glmb2.D93,do.coef=FALSE))
glmb.influence.measures(glmb2.D93,influence(glmb2.D93))


## Do not need method functions
dfbeta(glmb2.D93,influence(glmb2.D93))
dfbeta(glmb2.D93$fit,influence(glmb2.D93))
hatvalues(glmb2.D93,influence(glmb2.D93))
hatvalues(glmb2.D93$fit,influence(glmb2.D93))

# Methods (implmented)

rstandard(glmb2.D93,infl=influence(glmb2.D93))
rstandard(glmb2.D93)


# Need Metods

## Needs a method function
#dfbetas(glmb2.D93,influence(glmb2.D93))
```

```
dfbetas(glmb2.D93)
dfbetas(glmb2.D93,influence(glmb2.D93,do.coef=TRUE))
dfbetas(glmb2.D93$fit,influence(glmb2.D93))



# Needs a method function
cooks.distance(glmb2.D93)
cooks.distance(glmb2.D93$fit,influence(glmb2.D93))



# Needs a method function (now works)
#rstandard(glmb2.D93$fit,influence(glmb2.D93))


# Needs a method function
rstudent(glmb2.D93)
rstudent(glmb2.D93$fit,influence(glmb2.D93))


# Not a method, separate function
glmb.dffits(glmb2.D93)
dffits(glmb2.D93$fit,influence(glmb2.D93))

# Not a method - separate function
glmb.covratio(glmb2.D93)
covratio(glmb2.D93$fit,influence(glmb2.D93))



# Needs a method function but requires a different approach
# The influence function actually stores this measure
## This seems to require more work to create a method function
## Should
#hat(glmb2.D93,influence(glmb2.D93))
#hat(glmb2.D93$fit,influence(glmb2.D93))

infl=influence(glmb2.D93)
hat2=infl$hat
hat2
```

`Inv_f3_gaussian` *Derives the inverse of the gradient vector.*

Description

Computes the inverse of the gradient vector for the gaussian model. Typically done to find the set of tangency points that yield the same gradient as an initial set of gradients used for an envelope positioned at the posterior mode for a specific dispersion.

Usage

```
Inv_f3_gaussian(cbars, y, x, mu, P, alpha, wt)
```

Arguments

cbars	Gradient vectors desired for new thetabars. Typically the output of an initial call to Envelopebuild.
x, y	For glm: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For glm.fit: x is a design matrix of dimension n * p, and y is a vector of observations of length n.
mu	Prior mean
P	Prior Precision matrix
alpha	offset vector
wt	weighting vector

Value

Refer to Nygren and Nygren (2006) for details. The first set of items refers to Example 2 in section 3.1. All except the last item in this list of returned items has a number of rows equaling the number of components of the grid and a number of columns equaling the number of coefficients in the model. All quantities refer to the respective coefficient for each of the components of the grid.

Down	The lower bounds for the interval to be evaluated. Either negative infinity or a real number.
Up	The upper bounds for the interval to be evaluated. Either positive infinity or a real number.
lglt	The log of the density between negative infinity and the upper bound
lgrt	The log of the density between the lower bound and infinity
lgct	The log of the density between the lower and upper bounds
logU	The one of the 3 above that is relevant for the component of the grid
logP	A two column matrix, the first of which holds sum of logU across the components. The second column is 0 and is later populated by the Set_logP function

Examples

```
## ----dobson----------------------------------------------------------------
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)

## Prior mean vector
mu<-matrix(0,5)
mu[1,1]=log(mean(counts))
## Prior standard deviation and Variance
mysd<-1
V=((mysd)^2)*diag(5)
## Call to glmb
glmb.D93<-glmb(n=1000,counts ~ outcome + treatment,
               family = poisson(),pfamily=dNormal(mu=mu,Sigma=V))
## ----glmb extractAIC-------------------------------------------------------
extractAIC(glmb.D93)
```

lmb *Fitting Bayesian Linear Models*

Description

lmb is used to fit Bayesian linear models, specified by giving a symbolic descriptions of the linear predictor and the prior distribution.

Usage

```
lmb(
  formula,
  pfamily,
  n = 1000,
  data,
  subset,
  weights,
  na.action,
  method = "qr",
  model = TRUE,
  x = TRUE,
  y = TRUE,
  qr = TRUE,
  singular.ok = TRUE,
  contrasts = NULL,
  offset,
  ...
)

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

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
pfamily	a description of the prior distribution and associated constants to be used in the model. This should be a pfamily function (see pfamily for details of pfamily functions).
n	number of draws to generate. If length(n) > 1, the length is taken to be the number required.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which lm is called.
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, weighted least squares is used with weights weights (that is, minimizing sum(w*e^2)); otherwise ordinary least squares is used. See also 'Details',
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The 'factory-fresh' default is stats{na.omit}. Another possible value is NULL, no action. Value stats{na.exclude} can be useful.
method	the method to be used in fitting the classical model during a call to glm. The default method glm.fit uses iteratively reweighted least squares (IWLS): the alternative "model.frame" returns the model frame and does no fitting. User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If specified as a character string it is looked up from within the **stats** namespace.
model, x, y, qr	logicals. If TRUE the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.
singular.ok	logical. If FALSE (the default in S but not in R) a singular fit is an error.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
offset	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector or matrix of extents matching those of the response. One or more offset terms can be included in the formula instead or as well, and if more than one are specified their sum is used. See model.offset.
...	For lm(): additional arguments to be passed to the low level regression fitting functions (see below).
digits	the number of significant digits to use when printing.

Details

The function lmb is a Bayesian version of the classical lm function. Setup of the models mirrors those for the lm function but add the required argument pfamily with a prior formulation. The function generates random iid samples from the posterior density associated with the model.

The function returns the output from a call to the function lm as well as the simulated Bayesian coefficients and associated outputs. In addition, the function returns model diagnostic information related to the DIC, a Bayesian Information Criterion similar to the AIC for classical models.

For the gaussian family, iid samples from the posterior density are genererated using standard simulation procedures for multivariate normal densities. For all other families, the samples are generated using accept-reject procedures by leveraging the likelihood subgradient approach of Nygren and Nygren (2006). This approach relies on tight enveloping functions that bound the posterior density from above. The Gridtype parameter is used to select the method used for determining the size of a Grid used to build the enveloping function. See EnvelopeBuild for details. Depending on the selection, the time to build the envelope and the acceptance rate during the simulation process may vary. The returned value iters contains the number of candidates generated before acceptance for each draw.

Value

glmb returns an object of class "glmb". The function summary (i.e., summary.glmb) can be used to obtain or print a summary of the results. The generic accessor functions coefficients, fitted.values,

residuals, and extractAIC can be used to extract various useful features of the value returned by glmb.

An object of class "lmb" is a list containing at least the following components:

lm	an object of class "lm" containing the output from a call to the function lm
coefficients	a matrix of dimension n by length(mu) with one sample in each row
coef.means	a vector of length(mu) with the estimated posterior mean coefficients
coef.mode	a vector of length(mu) with the estimated posterior mode coefficients
dispersion	Either a constant provided as part of the call, or a vector of length n with one sample in each row.
Prior	A list with the priors specified for the model in question. Items in list may vary based on the type of prior
residuals	a matrix of dimension n by length(y) with one sample for the deviance residuals in each row
fitted.values	a matrix of dimension n by length(y) with one sample for the fitted values in each row
linear.predictors	an n by length(y) matrix with one sample for the linear fit on the link scale in each row
deviance	an n by 1 matrix with one sample for the deviance in each row
pD	An Estimate for the effective number of parameters
Dbar	Expected value for minus twice the log-likelihood function
Dthetabar	Value of minus twice the log-likelihood function evaluated at the mean value for the coefficients
DIC	Estimated Deviance Information criterion
weights	a vector of weights specified or implied by the model
prior.weights	a vector of weights specified or implied by the model
y	a vector of observations of length m.
x	a design matrix of dimension m * p
model	if requested (the default),the model frame
call	the matched call
formula	the formula supplied
terms	the terms object used
data	the data argument
famfunc	family functions used during estimation and post processing
iters	an n by 1 matrix giving the number of candidates generated before acceptance for each sample.
contrasts	(where relevant) the contrasts used.
xlevels	(where relevant) a record of the levels of the factors used in fitting
pfamily	the prior family specified
digits	the number of significant digits to use when printing.

In addition, non-empty fits will have (yet to be implemented) components qr, R and effects relating to the final weighted linear fit for the posterior mode. Objects of class "glmb" are normall of class c("glmb","glm","lm"), that is inherit from classes glm and lm and well-designed methods from those classed will be applied when appropriate.

If a binomial glmb model was specified by giving a two-column response, the weights returned by prior.weights are the total number of cases (factored by the supplied case weights) and the component of y of the result is the proportion of successes.

Author(s)

The R implementation of lmb has been written by Kjell Nygren and was built to be a Bayesian version of the lm function and hence tries to mirror the features of the lm function to the greatest extent possible while also taking advantage of some of the method functions developed for the glmb function. For details on the author(s) for the lm function see the documentation for lm.

References

Chambers, J.M.(1992) *Linear models.* Chapter 4 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

Wilkinson, G.N. and Rogers, C.E. (1973). Symbolic descriptions of factorial models for analysis of variance. *Applied Statistics*, **22**, 392-399. doi: 10.2307/2346786.

Nygren, K.N. and Nygren, L.M (2006) Likelihood Subgradient Densities. *Journal of the American Statistical Association*. vol.101, no.475, pp 1144-1156. doi: 10.1198/016214506000000357.

Raiffa, Howard and Schlaifer, R (1961) *Applied Statistical Decision Theory.* Boston: Clinton Press, Inc.

See Also

The classical modeling functions lm and glm.

pfamily for documentation of pfamily functions used to specify priors.

Prior_Setup, Prior_Check for functions used to initialize and to check priors,

summary.glmb, predict.glmb, simulate.glmb, extractAIC.glmb, dummy.coef.glmb and methods(class="glmb") for methods inherited from class glmb and the methods and generic functions for classes glm and lm from which class lmb also inherits.

Other glmbayes modeling functions: glmb(), rglmb(), rlmb()

Examples

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

ps=Prior_Setup(weight ~ group)
mu=ps$mu
V=ps$Sigma
mu[1,1]=mean(weight)

Prior_Check(weight ~ group,family =gaussian(),
           pfamily=dNormal(mu=mu,Sigma=V))
```

```
lm.D9 <- lm(weight ~ group,x=TRUE,y=TRUE)
disp_ML=sigma(lm.D9)^2
n_prior=2
shape=n_prior/2
rate= disp_ML*shape

# Conjugate Normal_Gamma Prior
lmb.D9=lmb(weight ~ group,dNormal_Gamma(mu,V/disp_ML,shape=shape,rate=rate))
summary(lmb.D9)

# Independent_Normal_Gamma_Prior
lmb.D9_v2=lmb(weight ~ group,dIndependent_Normal_Gamma(mu,V,shape=shape,rate=rate))
summary(lmb.D9_v2)
```

`logLik.glmb` *Extract Log-Likelihood*

Description

This function is a method function for the "glmb" class used to Extract the log-Likelihood from a Bayesian Generalized Linear Model.

Usage

```
## S3 method for class 'glmb'
logLik(object, ...)
```

Arguments

object	an object of class glmb, typically the result of a call to glmb
...	further arguments to or from other methods

Value

The function returns a vector, logLikout with the estimated log-likelihood for each draw.

Examples

```
## ----dobson----------------------------------------------------------------
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)

## Prior mean vector
mu<-matrix(0,5)
mu[1,1]=log(mean(counts))
## Prior standard deviation and Variance
mysd<-1
V=((mysd)^2)*diag(5)
## Call to glmb
glmb.D93<-glmb(n=1000,counts ~ outcome + treatment,
               family = poisson(),pfamily=dNormal(mu=mu,Sigma=V))
```

```
## ----glmb logLik-----------------------------------------------------------
colMeans(logLik(glmb.D93))
```

`menarche2` *Age Of Menarche In Warsaw*

Description

Proportions of female children at various ages during adolescence who have reached menarche.

Usage

```
data(menarche2)
```

Format

This data frame contains the following columns:

`Age` Average age of the group. (The groups are reasonably age homogeneous.)

`Total` Total number of children in the group.

`Menarche` Number who have reached menarche.

Details

Details to be added...

Source

MASS package (find more details).

References

Venables, W. N. and Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth edition. Springer.

Examples

```
data(menarche2)
```

Neg_logLik *Negative Log-Likelihood for a Generalized Linear Model*

Description

Negative Log-Likelihood for a Generalized Linear Model

Usage

```
Neg_logLik(b, y, x, alpha, wt, family)

Neg_logLik2(b, y, x, alpha, wt, family)
```

Arguments

b	a matrix where each row represents a point at which Negative Log-Likelihood is to be calculated
y	A vector
x	A matrix
alpha	A vector
wt	A vector
family	A family

Value

The sum of x and y

Examples

```
1+1
10+1
```

Normal_ct *The Central Normal Distribution*

Description

Distribution function and random generation for the center (between a lower and an upper bound) of the normal distribution with mean equal to mu and standard deviation equal to sigma.

Usage

```
pnorm_ct(a = -Inf, b = Inf, mu = 0, sigma = 1, log.p = TRUE, Diff = FALSE)

rnorm_ct(n, lgrt, lglt, mu = 0, sigma = 1)
```

Arguments

a	lower bound
b	upper bound
mu	mean parameter
sigma	standard deviation
log.p	Logical argument. If TRUE, the log probability is provided
Diff	Logical argument. If TRUE the second parameter is the difference between the lower and upper bound
n	number of draws to generate. If length(n) > 1, the length is taken to be the number required
lgrt	log of the distribution function between the lower bound and infinity
lglt	log of the distribution function between negative infinity and the upper bound

Details

The distribution function pnorm_ct finds the probability of the center of a normal density (the probability of the area between a lower bound a and an upper bound b) while the random number generator rnorm_ct samples from a restricted normal density where lgrt is the log of the distribution between the lower bound and infinity and lglt is the log of the distribution function between negative infinity and the upper bound. The sum of the exponentiated values for the two (exp(lgrt)+exp(lglt)) must sum to more than 1.

These functions are mainly used to handle cases where the differences between the upper and lower bounds b-a are small. In such cases, using pnorm(b)-pnorm(a) may result in 0 being returned even when the difference is supposed to be positive.

Value

For pnorm_ct, vector of length equal to length of a and for rnorm_ct, a vector with length determined by n containing draws from the center of the normal distribution.

Examples

```
pnorm_ct(0.2,0.4)
exp(pnorm_ct(0.2,0.4))
pnorm_ct(0.2,0.4,log.p=FALSE)
log(pnorm_ct(0.2,0.4,log.p=FALSE))
## Example where difference between two pnorm calls fail
## but call to pnorm_ct works
pnorm(0.5)-pnorm(0.4999999999999999)
pnorm_ct(0.4999999999999999,0.5,log.p=FALSE)
```

---

pfamily *Prior Family Objects for Bayesian Models*

Description

Prior family objects provide a convenient way to specify the details of the priors used by functions such as glmb. See the documentations for lmb, glmb, glmb, and rglmb for the details of how such model fitting takes place.

Usage

```
pfamily(object, ...)

dNormal(mu, Sigma, dispersion = NULL)

dGamma(shape, rate, beta)

dNormal_Gamma(mu, Sigma, shape, rate)

dIndependent_Normal_Gamma(mu, Sigma, shape, rate)

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

Arguments

object	the function pfamily accesses the pfamily objects which are stored within objects created by modelling functions (e.g., glmb).
mu	a prior mean vector for the the modeling coefficients used in several pfamilies
Sigma	a prior Variance-Covariance matrix for the model coefficients in several pfamilies
dispersion	the dispersion to be assumed when it is not given a prior. Should be provided when the Normal prior is for the gaussian(), Gamma(), quasibinomial, or quasipoisson families. The binomial() and poisson() families do not have dispersion coefficients.
shape	the prior shape parameter used by the gamma component of the prior. The gamma distribution is used as a prior for the inverse dispersion coefficients.
rate	the rate parameter used by the gamma component of the prior.
beta	the regression coefficients to be assumed when it is not given a prior. Needs to be provided when the Gamma prior is used for the dispersion. This specification is typically only used as part of Gibbs sampling where the beta and dispersion parameters are updated separately.
x	an object, a pfamily function that is to be printed
...	additional argument(s) for methods.

Details

pfamily is a generic function with methods for classe glmb and lmb. Many glmb models currently only have implementations for the dNormal() prior family. The Gamma() family also works with the dGamma() prior family while the gaussian() family works with the dGamma() and dNormal_Gamma() pfamilies.

Value

An object of class "pfamily" (which has a concise print method). This is a list with elements.

pfamily	character: the pfamily name
prior_list	a list with the prior parameters associated with the prior specification
okfamilies	currently implemented families for which the prior family can be used.

plinks	a function that assigns a set of oklinks for the combination of a family and and pfamily.
simfun	function: the function used to generate samples from the posterior density. All currently implemented pfamiles have simulation functions that generate iid samples for the associated posterior distribution.

Author(s)

The design of the pfamily set of functions was developed by Kjell Nygren and was inspired by the family used by the glmb function to specify the likelihood function. That design in turn was inspired by S functions of the same names described in Hastie and Pregibon (1992).

References

Cox, D.R. and Snell, E.J. (1981) *Applied Statistics; Principles and Examples.* London: chapman and Hall.

Dobson, A. J. (1990) *An Introduction to Statistical Modeling.* London: Chapman and Hall.

Hastie, T. J. and Pregibon, D. (1992) *Generalized linear models.* Chapter 6 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole. McCullagh P. and Nelder, J. A. (1989) *Generalized Linear Models.* London: Chapman and Hall.

Nygren, K.N. and Nygren, L.M (2006) Likelihood Subgradient Densities. *Journal of the American Statistical Association*. vol.101, no.475, pp 1144-1156. doi: 10.1198/016214506000000357.

Raiffa, Howard and Schlaifer, R (1961) *Applied Statistical Decision Theory.* Boston: Clinton Press, Inc.

See Also

lmb, glmb, rlmb, rglmb for modeling functions using pfamilies

rNormal_reg, rNormal_Gamma_reg, and rGamma_reg for lower level functions that sample from the resulting posterior distributions from the currently available pfamilies.

Examples

```
mu=c(0,0)
Sigma=diag(2)

npf<-dNormal(mu,Sigma)  # Normal pfamily
str(dNormal(mu,Sigma))

## Example where # Normal pfamily is used

## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
mysd<-1
mu<-matrix(0,5)
mu[1]=log(mean(counts))
V0<-((mysd)^2)*diag(5)
glmb.D93<-glmb(counts ~ outcome + treatment, family = poisson(),pfamily=dNormal(mu=mu,Sigma=V0))
summary(glmb.D93)
```

predict.glmb *Predict Method for Bayesian GLM Fits*

Description

Obtains predictions and optionally estimates standard errors of those predictions from a fitted Bayesian generalized linear model object.

Usage

```
## S3 method for class 'glmb'
predict(
  object,
  newdata = NULL,
  type = "link",
  se.fit = FALSE,
  dispersion = NULL,
  terms = NULL,
  na.action = na.pass,
  olddata = NULL,
  ...
)
```

Arguments

object	a fitted object of class inheriting from "glmb".
newdata	optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
type	the type of prediction required. The default is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable. Thus for a default binomial model the default predictions are of log-odds (probabilities on logit scale) and type = "response" gives the predicted probabilities. The "terms" option returns a matrix giving the fitted values of each term in the model formula on the linear predictor scale (not implemented).
se.fit	logical switch indicating if standard errors are required (not implemented).
dispersion	the dispersion of the Bayesian GLM fit to be assumed in computing the standard errors. If omitted, that returned by the summary applied to the object is used.
terms	with type="terms" by default all terms are returned. A character vector specifies which terms are to be returned.
na.action	function determining what should be done with missing values in newdata. The default is to predict NA.
olddata	a data frame that should contain all the variables used in the original model specification. Must currently be provided whenever newdata is provided. Both olddata and newdata are subsetted to the model variables extracte from the object model formula and rbind(olddata,newdata) must be valid after this step. A check is also run to verify if the resulting x matrix from olddata is consistent with that from the original model object.
...	further arguments passed to or from other methods.

Details

If newdata is omitted the predictions are based on the data used for the fit. In that case how cases with missing values in the original fit is determined by the na.action argument of that fit. If na.action = na.omit omitted cases will not appear in the residuals, whereas if na.action = na.exclude they will appear (in predictions and standard errors), with residual value NA. See also napredict.

Value

A matrix of predictions where the rows correspond to the draws from the estimated model, and the columns to the observations in the newdata dataset (or the original data if newdata is missing).

Note

Variables are first looked for in newdata and then searched for in the usual way (which will include the environment of the formula used in the fit). A warning will be given if the variables found are not of the same length as those in newdata if it was supplied.

Examples

```
data(menarche2)
## ----Analysis Setup--------------------------------------------------------
## Number of variables in model
Age=menarche2$Age
nvars=2
## Reference Ages for setting of priors and Age_Difference
ref_age1=13  # user can modify this
ref_age2=15  ## user can modify this
## Define variables used later in analysis
Age2=Age-ref_age1
Age_Diff=ref_age2-ref_age1
mu1=as.matrix(c(0,1.098612),ncol=1)
V1<-1*diag(nvars)
V1[1,1]=0.18687882
V1[2,2]=0.10576217
V1[1,2]=-0.03389182
V1[2,1]=-0.03389182
Menarche_Model_Data=data.frame(Age=menarche2$Age,Total=menarche2$Total,
                               Menarche=menarche2$Menarche,Age2)
glmb.out1<-glmb(n=1000,cbind(Menarche, Total-Menarche) ~Age2,family=binomial(logit),
                pfamily=dNormal(mu=mu1,Sigma=V1),data=Menarche_Model_Data)

# Prediction from original model
pred0=predict(glmb.out1,type="link")
colMeans(pred0)

pred1=predict(glmb.out1,type="response")
colMeans(pred1)


## Generate new data
Age_New <- seq(8, 20, 0.25)
Age2_New=Age_New-13
mod_Object=glmb.out1
obs_size=median(menarche2$Total) ## Counts for sim from Binomial
olddata=data.frame(Age=menarche2$Age,
```

```
                    Menarche=menarche2$Menarche,Total=menarche2$Total,Age2=Age2)

newdata=data.frame(Age=Age_New,Age2=Age2_New)

# Simulate for newdata

pred_menarche=predict(mod_Object,newdata=newdata,olddata=olddata,type="response")
pred_m=colMeans(pred_menarche)

n=nrow(mod_Object$coefficients)
pred_y=matrix(0,nrow=n,ncol=length(Age_New))
for(i in 1:n){
  pred_y[i,1:length(Age_New)]=rbinom(length(Age_New),size=obs_size,
  prob=pred_menarche[i,1:length(Age_New)
                                                                                ])
}

# Produce various predictions
pred_y_m=colMeans(pred_y/obs_size)
quant1_m=apply(pred_menarche,2,FUN=quantile,probs=c(0.025))
quant2_m=apply(pred_menarche,2,FUN=quantile,probs=c(0.975))
quant1_m_y=apply(pred_y/obs_size,2,FUN=quantile,probs=c(0.025))
quant2_m_y=apply(pred_y/obs_size,2,FUN=quantile,probs=c(0.975))

#Plot Predictions for newdata
plot(Menarche/Total ~ Age, data=menarche2,
main="Percentage of girls Who have had their first period")
lines(Age_New, pred_m,lty=1)
lines(Age_New, quant1_m,lty=2)
lines(Age_New, quant2_m,lty=2)
lines(Age_New, quant1_m_y,lty=2)
lines(Age_New, quant2_m_y,lty=2)
```

`Prior_Check` *Checks for Prior-data conflicts*

Description

Checks if the credible intervals for the prior overlaps with the implied confidence intervals from the classical model that comes from a call to the glm function

Usage

```
Prior_Check(
  formula,
  family,
  pfamily,
  level = 0.95,
  data = NULL,
  weights,
  subset,
  na.action,
  start = NULL,
```

```
  etastart,
  mustart,
  offset,
  control = list(...),
  model = TRUE,
  method = "glm.fit",
  x = FALSE,
  y = TRUE,
  contrasts = NULL,
  ...
)
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
family	a description of the error distribution and link function to be used in the model. For glm this can be a character string naming a family function, a family function or the result of a call to a family function. For glm.fit only the third option is supported. (See family for details of family functions.)
pfamily	a description of the prior distribution and associated constants to be used in the model. This should be a pfamily function (see pfamily for details of pfamily functions).
level	the confidence level at which the Prior-data conflict should be checked.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which glm is called.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The 'factory-fresh' default is stats{na.omit}. Another possible value is NULL, no action. Value stats{na.exclude} can be useful.
start	starting values for the parameters in the linear predictor.
etastart	starting values for the linear predictor.
mustart	starting values for the vector of means.
offset	this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See documentation for model.offset at model.extract.
control	a list of parameters for controlling the fitting process. For glm.fit this is passed to glm.control.
model	a logical value indicating whether *model frame* should be included as a component of the returned value.

method	the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS): the alternative "model.frame" returns the model frame and does no fitting. User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If specified as a character string it is looked up from within the **stats** namespace.
x, y	For glm: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For glm.fit: x is a design matrix of dimension n * p, and y is a vector of observations of length n.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
...	For glm: arguments to be used to form the default control argument if it is not supplied directly. For weights: further arguments passed to or from other methods.

Value

A vector where each item provided the ratio of the absolue value for the difference between the prior and maximum likelihood estimate divided by the length of the sum of half of the two intervals (where normality is assumed)

See Also

Other prior utility Functions: Prior_Setup()

Examples

```
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
print(d.AD <- data.frame(treatment, outcome, counts))
## Step 1: Set up Prior
ps=Prior_Setup(counts ~ outcome + treatment)
mu=ps$mu
V=ps$Sigma
# Step2A: Check the Prior
Prior_Check(counts ~ outcome + treatment,family = poisson(),
            pfamily=dNormal(mu=mu,Sigma=V))
# Step2B: Update and Re-Check the Prior
mu[1,1]=log(mean(counts))
Prior_Check(counts ~ outcome + treatment,family = poisson(),
            pfamily=dNormal(mu=mu,Sigma=V))
```

Prior_Setup *Setup Prior Objects*

Description

Sets up the structure for the Prior mean and Variance Matrices using information from a classical model.

Usage

```
Prior_Setup(
  formula,
  data = NULL,
  subset = NULL,
  na.action = na.fail,
  drop.unused.levels = FALSE,
  xlev = NULL,
  ...
)
```

Arguments

formula	a model formula or terms object or an R object.
data	a data frame, list or environment (or object coercible by as.data.frame to a data frame), containing the variables in formula. Neither a matrix nor an array will be accepted.
subset	a specification of the rows/observations to be used: defaults to all. This can be any valid indexing vector (see [.data.frame) for the rows of data, or a (logical) expression using variables in data or if that is not supplied, in formula. (See additional details about how this argument interacts with data-dependent bases under 'Details' below.)
na.action	how NAs are treated. The default is first, any na.action attribute of data, second a na.action setting of options, and third na.fail if that is unset. The factory-fresh default is na.omit. Another possible value is NULL.
drop.unused.levels	should factors have unused levels dropped? Defaults to FALSE.
xlev	a named list of character vectors giving the full set of levels to be assumed for each factor.
...	for model.frame methods, a mix of further arguments such as data, na.action, subset to pass to the default method. Any additional arguments (such as offset and weights or other named arguments) which reach the default method are used to create further columns in the model frame, with parenthesised names such as "(offset)". For get_all_vars, further named columns to include in the model frame.

Value

A list with items related to the prior.

mu	An initial version of the prior mean vector, populated with all zeros

Sigma	An initial version of the prior Variance-Covariance vector, populated as the diagonal identity matrix
model	The model frame from object if it exists
x	The design matrix from object if it exists

See Also

Other prior utility Functions: Prior_Check()

Examples

```
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
print(d.AD <- data.frame(treatment, outcome, counts))
## Step 1: Set up Prior
ps=Prior_Setup(counts ~ outcome + treatment)
mu=ps$mu
V=ps$Sigma
print(ps)
```

---

residuals.glmb *Accessing Bayesian Generalized Linear Model Fits*

Description

These functions are all methods for class glmb or summary.glmb objects.

Usage

```
## S3 method for class 'glmb'
residuals(object, ysim = NULL, ...)

## S3 method for class 'lmb'
residuals(object, ysim = NULL, ...)
```

Arguments

object	an object of class glmb, typically the result of a call to glmb
ysim	Optional simulated data for the data y.
...	further arguments to or from other methods

Value

A matrix DevRes of dimension n times p containing the Deviance residuals for each draw. If ysim is provided, the residuals are based on a comparison to the simulated data instead. The credible intervals for residuals based on simulated data should be a more appropriate measure of whether individual residuals represent outliers or not.

Examples

```
data(menarche2)
## ----Analysis Setup--------------------------------------------------------
## Number of variables in model
Age=menarche2$Age
nvars=2
## Reference Ages for setting of priors and Age_Difference
ref_age1=13  # user can modify this
ref_age2=15  ## user can modify this
## Define variables used later in analysis
Age2=menarche2$Age-ref_age1
Age_Diff=ref_age2-ref_age1
mu1=as.matrix(c(0,1.098612),ncol=1)
V1<-1*diag(nvars)
V1[1,1]=0.18687882
V1[2,2]=0.10576217
V1[1,2]=-0.03389182
V1[2,1]=-0.03389182
Menarche_Model_Data=data.frame(Age=menarche2$Age,Total=menarche2$Total,
                               Menarche=menarche2$Menarche,Age2)
glmb.out1<-glmb(n=1000,cbind(Menarche, Total-Menarche) ~Age2,family=binomial(logit),
                pfamily=dNormal(mu=mu1,Sigma=V1),data=Menarche_Model_Data)

# Prediction from original model
pred1=predict(glmb.out1,type="response")

## Get Original Residuals, their means, and credible bounds
res_out=residuals(glmb.out1)
colMeans(res_out)

## Set up to simulate new data and residuals
res_mean=colMeans(res_out)
res_low1=apply(res_out,2,FUN=quantile,probs=c(0.025))
res_high1=apply(res_out,2,FUN=quantile,probs=c(0.975))

## Simulate new data and get residuals for simulated data

ysim1=simulate(glmb.out1,nsim=1,seed=NULL,pred=pred1,family="binomial",
               prior.weights=weights(glmb.out1))


res_ysim_out1=residuals(glmb.out1,ysim=ysim1)
res_low=apply(res_ysim_out1,2,FUN=quantile,probs=c(0.025))
res_high=apply(res_ysim_out1,2,FUN=quantile,probs=c(0.975))

# Plot Credible Interval bounds for Deviance Residuals

plot(res_mean~Age,ylim=c(-2.5,2.5),
main="Credible Interval Bound for Menarche - Logit Model Deviance Residuals",
xlab = "Age", ylab = "Avg. Dev. Res")
lines(Age, 0*res_mean,lty=1)
lines(Age, res_low,lty=1)
lines(Age, res_high,lty=1)
lines(Age, res_low1,lty=2)
lines(Age, res_high1,lty=2)
```

rGamma_reg *The Bayesian Generalized Linear Model Dispersion Distribution*

Description

rGamma_reg is used to generate iid samples for the dispersion parameter in Bayesian Generalized Linear models. The model is specified by providing the model structure, regression coefficient, and a Gamma prior.

Usage

```
rGamma_reg(
  n,
  y,
  x,
  prior_list,
  offset = NULL,
  weights = 1,
  family = gaussian()
)

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

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

Arguments

n	number of draws to generate. If length(n) > 1, the length is taken to be the number required.
y	a vector of observations of length m.
x	a design matrix of dimension m * p.
prior_list	a list with the prior parameters (shape and rate) for the prior distribution and a vector with the assumed value (beta) for the regression coefficients.
offset	this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
family	a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.)
digits	Number of significant digits to use for printed outpute
object	an object of class "glmb_dispersion" that is to be summarized
...	further arguments passed to or from other methods

Details

The rGamma_reg function produces iid samples for the dispersion parameter in Bayesian generalized linear models (gaussian and Gamma families only). Core required inputs for the function include the data vector, the design matrix, an estimate for the regression coefficients, and a prior gamma distribution specification. The function returns the simulated Bayesian conditional dispersion and the prior specification. The most practical use of this function is likely to be in conjunction with the rglmb function as part of block Gibbs sampling procedues.

For the gaussian family, iid samples from the posterior density are genererated using standard simulation procedures for gamma distributions. For the Gamma family, , the samples are generated using accept-reject procedures by leveraging the likelihood subgradient approach of Nygren and Nygren (2006).

Value

rGamma_reg returns a object of class "rglmbdisp". The function summary (i.e., summary.rglmb) can be used to obtain or print a summary of the results. The generic accessor functions coefficients, fitted.values, residuals, and extractAIC can be used to extract various useful features of the value returned by rGamma_reg. An object of class "rglmbdisp" is a list containing at least the following components:

dispersion	an n by 1 matrix with simulated values for the dispersion
Prior	A list with two components. The first being the prior mean vector and the second the prior precision matrix

References

A reference

See Also

Other family simulation functions: rNormal_Gamma_reg(), rNormal_reg(), rindependent_norm_gamma_reg()

Examples

```
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group,x=TRUE,y=TRUE)

lm_summary=summary(lm.D9)

lm_summary
lm.D9$coefficients

n<-10000
y<-lm.D9$y
x<-as.matrix(lm.D9$x)

### Set old regression and dispersion coefficients

b_old=lm.D9$coefficients
```

```
v_old=lm_summary$sigma^2

#### Set up for rglmb_dispersion

n0=0.1
shape=n0/2
rate=shape*v_old

rate/(shape-1)
rate/shape

#a1<-shape+n1/2
#b1<-rate+sum(SS)/2

#out<-1/rgamma(n,shape=a1,rate=b1)
## v0=sum(SS)/n1 ~ (2*rate+sum(SS))/(2*shape+n1)=[(2*rate+sum(SS))/2]/((2*shape+n1)/2)
n1=length(y)
SS=v_old*n1

n1 # This is equal to 20


n0=2 # Prior observations
v_prior=v_old  # Prior point estimate for variance (the mean of (1/dispersion=1/v_prior))
wt0=(n0/n1)

## set shape=0.01*(n1/2)
## set rate= 0.01*SS/2

shape=wt0*(n1/2)   ###  Shape is prior observations /2
rate=shape*v_prior  ### rate is essentiall prior SS - V in rmultireg should be this
rate/shape ## Should match v_prior (currently also v_old)

## We see that this currently matches
### (test different v_prior with various prior observations below)

prior_list=list(beta=b_old,shape=shape,rate=rate)

dispout<-rGamma_reg(n=n,y,x,prior_list=prior_list,
offset= rep(0, length(y)),family=gaussian())

mean(dispout$dispersion)
v_prior
v_old
#summary(dispout)  ## Summary function not working - need to write

### Set up test rglmb regressions without and with prior for dispersion

mu<-c(0,0)
mu=b_old  ### For testing purposes, set prior=b_old
P<-0.1*diag(2)
wt2=rep(1,length(y))

### Check
prior=list(mu=mu,Sigma=solve(P),dispersion=v_old)
outtemp1<-glmb(n = 1000, weight ~ group, family = gaussian(),
pfamily=dNormal(mu=mu,Sigma=solve(P),dispersion=v_old))
```

```
## Could use a residuals function here -- For now, maybe run the glmb function

summary(outtemp1)
mean(colMeans(residuals(outtemp1)^2))
v_old
colMeans((outtemp1$coefficients))
b_old
prior=list(mu=mu,P=P,dispersion=v_old)
outtemp2<-rglmb(n = 1000, y, x, family = gaussian(),
pfamily=dNormal(mu=mu,Sigma=solve(P),dispersion=v_old),
offset = rep(0, length(y)), weights = wt2)
summary(outtemp2)
colMeans((outtemp2$coefficients))
b_old

prior=list(mu=mu,Sigma=solve(P),shape=shape,rate=rate)
outtemp3<-glmb(n = 10000, weight ~ group,family = gaussian(),
               dNormal_Gamma(mu=mu,Sigma=solve(P),shape=shape,rate=rate))
## Could use a residuals function here -- For now, maybe run the glmb function

summary(outtemp3)
mean(colMeans(residuals(outtemp3)^2))
v_old
colMeans((outtemp3$coefficients))
b_old
mean(outtemp3$dispersion)  ## Seems slightly smaller --> Needs qc
v_old
```

---

rglmb *The Bayesian Generalized Linear Model Distribution*

Description

rglmb is used to generate iid samples for Bayesian Generalized Linear Models. The model is specified by providing a data vector, a design matrix, the family (determining the likelihood function) and the pfamily (determining the prior distribution).

Usage

```
rglmb(n = 1, y, x, family = gaussian(), pfamily, offset = NULL, weights = 1)

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

Arguments

n	number of draws to generate. If length(n) > 1, the length is taken to be the number required.
y	a vector of observations of length m.
x	for rglmb a design matrix of dimension m * p and for print.rglmb the object to be printed.

family	a description of the error distribution and link function to be used in the model. For glm this can be a character string naming a family function, a family function or the result of a call to a family function. For glm.fit only the third option is supported. (See family for details of family functions.)
pfamily	a description of the prior distribution and associated constants to be used in the model. This should be a pfamily function (see pfamily for details of pfamily functions).
offset	this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See documentation for model.offset at model.extract.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
digits	the number of significant digits to use when printing.
...	For glm: arguments to be used to form the default control argument if it is not supplied directly. For weights: further arguments passed to or from other methods.

Value

rglmb returns a object of class "rglmb". The function summary (i.e., summary.rglmb) can be used to obtain or print a summary of the results. The generic accessor functions coefficients, fitted.values, residuals, and extractAIC can be used to extract various useful features of the value returned by rglmb. An object of class "rglmb" is a list containing at least the following components:

coefficients	a matrix of dimension n by length(mu) with one sample in each row
coef.mode	a vector of length(mu) with the estimated posterior mode coefficients
dispersion	Either a constant provided as part of the call, or a vector of length n with one sample in each row.
Prior	A list with the priors specified for the model in question. Items in the list may vary based on the type of prior
prior.weights	a vector of weights specified or implied by the model
y	a vector with the dependent variable
x	a matrix with the implied design matrix for the model
famfunc	Family functions used during estimation process
iters	an n by 1 matrix giving the number of candidates generated before acceptance for each sample.
Envelope	the envelope that was used during sampling

Author(s)

The R implementation of rglmb has been written by Kjell Nygren and was built to be a Bayesian version of the glm function but with a more minimalistic interface than the glmb function. It also borrows some of its structure from other random generating function like rnorm and hence the r prefix.

References

Dobson, A. J. (1990) *An Introduction to Generalized Linear Models.* London: Chapman and Hall.

Hastie, T. J. and Pregibon, D. (1992) *Generalized linear models.* Chapter 6 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole. McCullagh P. and Nelder, J. A. (1989) *Generalized Linear Models.* London: Chapman and Hall.

Nygren, K.N. and Nygren, L.M (2006) Likelihood Subgradient Densities. *Journal of the American Statistical Association*. vol.101, no.475, pp 1144-1156. doi: 10.1198/016214506000000357.

Raiffa, Howard and Schlaifer, R (1961) *Applied Statistical Decision Theory.* Boston: Clinton Press, Inc.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* New York: Springer.

See Also

`lm` and `glm` for classical modeling functions.

`family` for documentation of family functions used to specify priors. `pfamily` for documentation of pfamily functions used to specify priors.

`Prior_Setup`, `Prior_Check` for functions used to initialize and to check priors,

`summary.glmb`, `predict.glmb`, `residuals.glmb`, `simulate.glmb`, `extractAIC.glmb`, `dummy.coef.glmb` and methods(class="glmb") for `glmb` and the methods and generic functions for classes `glm` and `lm` from which class `glmb` inherits.

Other glmbayes modeling functions: `glmb()`, `lmb()`, `rlmb()`

Examples

```
data(menarche2)

summary(menarche2)
plot(Menarche/Total ~ Age, data=menarche2)

Age2=menarche2$Age-13

x<-matrix(as.numeric(1.0),nrow=length(Age2),ncol=2)
x[,2]=Age2

y=menarche2$Menarche/menarche2$Total
wt=menarche2$Total

mu<-matrix(as.numeric(0.0),nrow=2,ncol=1)
mu[2,1]=(log(0.9/0.1)-log(0.5/0.5))/3

V1<-1*diag(as.numeric(2.0))

# 2 standard deviations for prior estimate at age 13 between 0.1 and 0.9
## Specifies uncertainty around the point estimates

V1[1,1]<-((log(0.9/0.1)-log(0.5/0.5))/2)^2
V1[2,2]=(3*mu[2,1]/2)^2  # Allows slope to be up to 3 times as large as point estimate

out<-rglmb(n = 1000, y=y, x=x, pfamily=dNormal(mu=mu,Sigma=V1), weights = wt,
           family = binomial(logit))
summary(out)
```

```
# Add mean(out$iters to rglmb summary function)
mean(out$iters)
```

---

rindependent_norm_gamma_reg *The Bayesian Indendepent Normal-Gamma Regression Distribution*

Description

Generates iid samples from the posterior density for the independent normal-gamma regression

Usage

```
rindependent_norm_gamma_reg(
  n,
  y,
  x,
  prior_list,
  offset = NULL,
  weights = 1,
  family = gaussian(),
  max_disp_perc = 0.99
)

rindependent_norm_gamma_reg_v3(
  n,
  y,
  x,
  prior_list,
  offset = NULL,
  weights = 1,
  family = gaussian(),
  max_disp_perc = 0.99
)

rindependent_norm_gamma_reg_v2(
  n,
  y,
  x,
  prior_list,
  offset = NULL,
  weights = 1,
  family = gaussian(),
  max_disp_perc = 0.99
)

rindependent_norm_gamma_reg_v4(
  n,
  y,
  x,
  prior_list,
```

```
  offset = NULL,
  weights = 1,
  family = gaussian(),
  max_disp_perc = 0.99
)
```

Arguments

n	number of draws to generate. If length(n) > 1, the length is taken to be the number required.
x, y	For glm: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For glm.fit: x is a design matrix of dimension n * p, and y is a vector of observations of length n.
prior_list	a list with the prior parameters (mu, Sigma, shape and rate) for the prior distribution.
offset	an offset parameter
weights	a weighting variable
family	a description of the error distribution and link function to be used in the model. For glm this can be a character string naming a family function, a family function or the result of a call to a family function. For glm.fit only the third option is supported. (See family for details of family functions.)
max_disp_perc	parameter currently used to control upper bound in accept-reject procedure

Details

The rindependent_norm_gamma_reg function produces iid samples for the Bayesian the Bayesian Independent Normal-Gamma Regression Distribution associated with the gaussian() family.

As this is a non-conjugate prior specification with a random dispersion parameter, we leverage methods that build on those developed for log-concave likelihood functions with a fixed dispersion parameter. A number of steps are first used in order to position an initial enveloping function for the main regression coefficients near the center of the posterior density and simultaneoulsy finding a modified Gamma distribution from which to generate candidate samples for the inverse dispersion.

During the simulation procedure, candidates are generated independently from the modified gamma distribution and a specific likelihood subgradient density (corresponding to the center of the density). A series of bounding functions are then used during the simulation process to determine if specific candidates should be accepted as follows

1) Similar to the procedure in the fixed dispersion case, the (modified) log-likelihood function is bounded using the value of the log-likelihood function at a specific point (which now depends on the dispersion and is picked to ensure the gradient vector is constant across simulated dispersion candidates) and the gradient at that value (which is kept fixed for each component in the grid regardless of dispersion.

2) A portion of the bounding function above is in turn bounded by using a term that has been shifted to rate parameter of the modified gamma candidate generating distribution.

3) Because candidates

matrix and a prior specification. The function returns the simulated Bayesian coefficients and some associated outputs. The iid samples from the posterior density is genererated using standard simulation procedures for multivariate normal and gamma distributions.

Value

rindependent_norm_gamma_reg returns a object of class "rglmb". The function summary (i.e., summary.rglmb) can be used to obtain or print a summary of the results. The generic accessor functions coefficients, fitted.values, residuals, and extractAIC can be used to extract various useful features of the value returned by rNormal_Gamma_reg. An object of class "rglmb" is a list containing at least the following components:

coefficients	a n by length(mu) matrix with one sample in each row
PostMode	a vector of length(mu) with the estimated posterior mode coefficients
Prior	A list with two components. The first being the prior mean vector and the second the prior precision matrix
iters	an n by 1 matrix giving the number of candidates generated before acceptance for each sample.
famfunc	an object of class "famfunc"
Envelope	an object of class "envelope"
dispersion	an n by 1 matrix with simulated values for the dispersion
loglike	a n by 1 matrix containing the negative loglikelihood for each sample.

See Also

Other family simulation functions: rGamma_reg(), rNormal_Gamma_reg(), rNormal_reg()

Examples

```
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group,y=TRUE,x=TRUE)

p_setup=Prior_Setup(lm.D9)
dispersion=sigma(lm.D9)^2

## Initialize dispersion
p=1/dispersion

mu=p_setup$mu
Sigma_prior=p_setup$Sigma
y=lm.D9$y
x=lm.D9$x

Prior_Check(weight ~ group,gaussian(),dNormal(mu=mu,Sigma=Sigma_prior))

## This simple "standardization" for the prior appears to
## correct the issue with the prior

## "Standardize" the prior for the intercept by setting the
## mean equal to the mean of the dependent variable
## and setting the variance equal to the variance of the dependent variable

mu[1,1]=mean(y)
```

```
Sigma_prior[1,1]=var(y)

## For factors, set the "standad prior to mean=0 and variance driven by the range
## of the dependent variable

mu[2,1]=0
Sigma_prior[2,2]=((max(y)-min(y))/1.96)^2

#Prior_Check(lm.D9,mu,Sigma_prior)
Prior_Check(weight ~ group,gaussian(),dNormal(mu=mu,Sigma=Sigma_prior))


#prior=list(mu=mu,Sigma=Sigma_prior,dispersion=dispersion)
glmb.D9=glmb(weight~group, family=gaussian(),dNormal(mu=mu,Sigma=Sigma_prior,dispersion=dispersion))
post_mode=glmb.D9$coef.mode

sum_out1=summary(glmb.D9)

## Try mean one standard deviation away to see how it works
#mu[1,1]=mu[1,1]-2*sum_out1$coefficients[1,3]
#mu[2,1]=mu[2,1]+2*sum_out1$coefficients[1,3]

#prior=list(mu=mu,Sigma=Sigma_prior,dispersion=dispersion)

# Temporarily lower the prior variance
Sigma_prior=1*Sigma_prior

glmb.D9_v2=glmb(n=1000,weight~group, family=gaussian(),
dNormal(mu=mu,Sigma=Sigma_prior,dispersion=dispersion))

n_prior=2
shape=n_prior/2
rate= dispersion*shape


summary(glmb.D9)
summary(glmb.D9_v2)


#lm_out=lm(y ~ x-1) # returns same model
RSS=sum(residuals(lm.D9)^2)

n_prior=4
n_data=length(y)
shape=(n_prior/2)
rate=n_prior*RSS/n_data



prior_list=list(mu=mu,Sigma=Sigma_prior,dispersion=dispersion,
                 shape=shape,rate=rate,Precision=solve(Sigma_prior))


set.seed(360)

 ptm <- proc.time()
 sim2=rindependent_norm_gamma_reg(n=1000,y,x,prior_list=prior_list,
```

```
offset=NULL,weights=1,max_disp_perc=0.99)
 proc.time()-ptm
```

---

rindep_norm_gamma_reg_std_R *The Standard Bayesian Indendepent Normal-Gamma Regression Distribution*

Description

Generates iid samples from the posterior density for the standard independent normal-gamma regression

Usage

```
rindep_norm_gamma_reg_std_R(
  n,
  y,
  x,
  mu,
  P,
  alpha,
  wt,
  f2,
  Envelope,
  gamma_list,
  UB_list,
  family,
  link,
  progbar = 1L
)
```

Arguments

n	number of draws to generate. If length(n) > 1, the length is taken to be the number required.
x, y	For glm: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For glm.fit: x is a design matrix of dimension n * p, and y is a vector of observations of length n.
mu	Prior mean
P	Prior Precision
alpha	offset

wt	weights
f2	Gradient function
Envelope	Envelope object
gamma_list	list with parameters used by to generate candidates from a restricted gamma
UB_list	list with constants used during accept-reject calculations.
family	a description of the error distribution and link function to be used in the model. For glm this can be a character string naming a family function, a family function or the result of a call to a family function. For glm.fit only the third option is supported. (See family for details of family functions.)
link	link function
progbar	progress bar

Details

The rindependent_norm_gamma_reg function produces iid samples for the Bayesian the Standardized Bayesian Independent Normal-Gamma Regression Distribution associated with the gaussian() family.

This generates candidates independently from a gamma distribution and a likelihood subgradient density and then uses an accept-reject procedure in order to get samples from the desired posterior density.

A series of bounding functions are used during the simulation process in order to bound the likelihood function associated with the posterior density

1) Similar to the procedure in the fixed dispersion case, the (modified) log-likelihood function is for a specific component of the grid is bounded

2) A portion of the bounding function above is in turn bounded by using a term that has been shifted to the rate parameter of the modified gamma candidate generating distribution. This term consist of the *RSS* associated with the maximum likelihood estimate.

3) Because the relative proportions of the

Value

Currently mainly the draws for the dispersion and the regression coefficients will be updated to return outputs consistent with other function

Examples

```
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group,y=TRUE,x=TRUE)

p_setup=Prior_Setup(lm.D9)
dispersion=sigma(lm.D9)^2

## Initialize dispersion
p=1/dispersion

mu=p_setup$mu
```

```
Sigma_prior=p_setup$Sigma
y=lm.D9$y
x=lm.D9$x

Prior_Check(weight ~ group,gaussian(),dNormal(mu=mu,Sigma=Sigma_prior))

## This simple "standardization" for the prior appears to
## correct the issue with the prior

## "Standardize" the prior for the intercept by setting the
## mean equal to the mean of the dependent variable
## and setting the variance equal to the variance of the dependent variable

mu[1,1]=mean(y)
Sigma_prior[1,1]=var(y)

## For factors, set the "standad prior to mean=0 and variance driven by the range
## of the dependent variable

mu[2,1]=0
Sigma_prior[2,2]=((max(y)-min(y))/1.96)^2

#Prior_Check(lm.D9,mu,Sigma_prior)
Prior_Check(weight ~ group,gaussian(),dNormal(mu=mu,Sigma=Sigma_prior))


#prior=list(mu=mu,Sigma=Sigma_prior,dispersion=dispersion)
glmb.D9=glmb(weight~group, family=gaussian(),dNormal(mu=mu,Sigma=Sigma_prior,dispersion=dispersion))
post_mode=glmb.D9$coef.mode

sum_out1=summary(glmb.D9)

## Try mean one standard deviation away to see how it works
#mu[1,1]=mu[1,1]-2*sum_out1$coefficients[1,3]
#mu[2,1]=mu[2,1]+2*sum_out1$coefficients[1,3]

#prior=list(mu=mu,Sigma=Sigma_prior,dispersion=dispersion)

# Temporarily lower the prior variance
Sigma_prior=1*Sigma_prior

glmb.D9_v2=glmb(n=1000,weight~group, family=gaussian(),
dNormal(mu=mu,Sigma=Sigma_prior,dispersion=dispersion))

n_prior=2
shape=n_prior/2
rate= dispersion*shape


summary(glmb.D9)
summary(glmb.D9_v2)


#lm_out=lm(y ~ x-1) # returns same model
RSS=sum(residuals(lm.D9)^2)

n_prior=4
```

```
n_data=length(y)
shape=(n_prior/2)
rate=n_prior*RSS/n_data



prior_list=list(mu=mu,Sigma=Sigma_prior,dispersion=dispersion,
                 shape=shape,rate=rate,Precision=solve(Sigma_prior))


set.seed(360)

 ptm <- proc.time()
 sim2=rindependent_norm_gamma_reg(n=1000,y,x,prior_list=prior_list,
offset=NULL,weights=1,max_disp_perc=0.99)
 proc.time()-ptm
```

`rlmb` *The Bayesian Linear Model Distribution*

Description

`rlmb` is used to generate iid samples from Bayesian Linear Models with multivariate normal priors. The model is specified by providing a data vector, a design matrix, and a pfamily (determining the prior distribution).

Usage

```
rlmb(n = 1, y, x, pfamily, offset = rep(0, nobs), weights = NULL)

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

Arguments

`n`	number of draws to generate. If `length(n) > 1`, the length is taken to be the number required.
`y`	a vector of observations of length `m`.
`x`	for `rlmb` a design matrix of dimension `m * p` and for `print.rlmb` the object to be printed.
`pfamily`	a description of the prior distribution and associated constants to be used in the model. This should be a pfamily function (see `pfamily` for details of pfamily functions.)

offset	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector or matrix of extents matching those of the response. One or more offset terms can be included in the formula instead or as well, and if more than one are specified their sum is used. See model.offset.
weights	an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector. If non-NULL, weighted least squares is used with weights weights (that is, minimizing sum(w*e^2)); otherwise ordinary least squares is used. See also 'Details',
digits	the number of significant digits to use when printing.
...	For lm(): additional arguments to be passed to the low level regression fitting functions (see below).

Details

The rlmb function produces iid samples for Bayesian generalized linear models. It has a more minimialistic interface than than the lmb function. Core required inputs for the function include the data vector, the design matrix and a prior specification. In addition, the dispersion parameter must currently be provided for the gaussian, Gamma, quasipoisson, and quasibinomial families (future implementations may incorporate a prior for these into the rlmb function).

Current implemented models include the gaussian family (identity link function), the poisson and quasipoisson families (log link function), the gamma family (log link function), as well as the binomial and quasibinomial families (logit, probit, and cloglog link functions). The function returns the simulated Bayesian coefficients and some associated outputs.

For the gaussian family, iid samples from the posterior density are genererated using standard simulation procedures for multivariate normal densities. For all other families, the samples are generated using accept-reject procedures by leveraging the likelihood subgradient approach of Nygren and Nygren (2006). This approach relies on tight enveloping functions that bound the posterior density from above. The Gridtype parameter is used to select the method used for determining the size of a Grid used to build the enveloping function. See EnvelopeBuild for details. Depending on the selection, the time to build the envelope and the acceptance rate during the simulation process may vary. The returned value iters contains the number of candidates generated before acceptance for each draw.

Value

rlmb returns a object of class "rlmb". The function summary (i.e., summary.rglmb) can be used to obtain or print a summary of the results. The generic accessor functions coefficients, fitted.values, residuals, and extractAIC can be used to extract various useful features of the value returned by rglmb. An object of class "rlmb" is a list containing at least the following components:

coefficients	a matrix of dimension n by length(mu) with one sample in each row
coef.mode	a vector of length(mu) with the estimated posterior mode coefficients
dispersion	Either a constant provided as part of the call, or a vector of length n with one sample in each row.
Prior	A list with the priors specified for the model in question. Items in the list may vary based on the type of prior
prior.weights	a vector of weights specified or implied by the model
y	a vector with the dependent variable
x	a matrix with the implied design matrix for the model

famfunc	Family functions used during estimation process
iters	an n by 1 matrix giving the number of candidates generated before acceptance for each sample.
Envelope	the envelope that was used during sampling

References

Chambers, J.M.(1992) *Linear models.* Chapter 4 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

Wilkinson, G.N. and Rogers, C.E. (1973). Symbolic descriptions of factorial models for analysis of variance. *Applied Statistics*, **22**, 392-399. doi: 10.2307/2346786.

Nygren, K.N. and Nygren, L.M (2006) Likelihood Subgradient Densities. *Journal of the American Statistical Association*. vol.101, no.475, pp 1144-1156. doi: 10.1198/016214506000000357.

Raiffa, Howard and Schlaifer, R (1961) *Applied Statistical Decision Theory.* Boston: Clinton Press, Inc.

See Also

The classical modeling functions `lm` and `glm`.

`pfamily` for documentation of pfamily functions used to specify priors.

`Prior_Setup`, `Prior_Check` for functions used to initialize and to check priors,

`summary.glmb`, `predict.glmb`, `simulate.glmb`, `extractAIC.glmb`, `dummy.coef.glmb` and methods(class="glmb") for methods inherited from class `glmb` and the methods and generic functions for classes `glm` and `lm` from which class `lmb` also inherits.

Other glmbayes modeling functions: `glmb()`, `lmb()`, `rglmb()`

Examples

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

ps=Prior_Setup(weight ~ group)
x=ps$x
mu=ps$mu
V=ps$Sigma
mu[1,1]=mean(weight)

Prior_Check(weight ~ group,family =gaussian(),
            pfamily=dNormal(mu=mu,Sigma=V))
lm.D9 <- lm(weight ~ group,x=TRUE,y=TRUE)
y=lm.D9$y

## Dispersion for maximum likelihood estimate
disp_ML=sigma(lm.D9)^2
n_prior=2
shape=n_prior/2
rate= disp_ML*shape
```

```
# Two-Block Gibbs sampler
set.seed(180)
dispersion2=disp_ML
# Run 1000 burn-in iterations
for(i in 1:1000){
  out1=rlmb(n = 1, y=y, x=x, pfamily=dNormal(mu=mu,Sigma=V,dispersion=dispersion2))
 out2=rlmb(n = 1, y=y, x=x, pfamily=dGamma(shape=shape,rate=rate,beta=out1$coefficients[1,]))
  dispersions=out2$dispersion
}


# Run 1000 additional iterations and store output
beta_out<-matrix(0,nrow=1000, ncol=2)
disp_out=rep(0,1000)
for(i in 1:1000){
  out1=rlmb(n = 1, y=y, x=x, pfamily=dNormal(mu=mu,Sigma=V,dispersion=dispersion2))
 out2=rlmb(n = 1, y=y, x=x, pfamily=dGamma(shape=shape,rate=rate,beta=out1$coefficients[1,]))
  dispersions=out2$dispersion
  beta_out[i,1:2]=out1$coefficients[1,1:2]
  disp_out[i]=out2$dispersion
}

colMeans(beta_out)
mean(disp_out)

# Same model using Independent_Normal_Gamma_Prior
lmb.D9_v2=lmb(weight ~ group,dIndependent_Normal_Gamma(mu,V,shape=shape,rate=rate))
summary(lmb.D9_v2)
```

`rnnorm_reg_std` *The Bayesian Generalized Linear Model Distribution in Standard Form*

Description

`rnnorm_reg_std` is used to generate iid samplers from Non-Gaussian Generalized Linear Models in standard form. The function should onlybe called after standardization of a Generalized Linear Model.

Usage

```
rnnorm_reg_std(
  n,
  y,
  x,
  mu,
  P,
  alpha,
  wt,
  f2,
  Envelope,
  family,
```

```
  link,
  progbar = 1L
)
```

Arguments

n	number of draws to generate. If length(n) > 1, the length is taken to be the number required.
y	a vector of observations of length m.
x	a design matrix of dimension m * p.
mu	a vector of length p giving the prior means of the variables in the design matrix.
P	a positive-definite symmetric matrix of dimension p * p specifying the prior precision matrix of the variable.
alpha	this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
wt	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
f2	function used to calculate the negative of the log-posterior function
Envelope	an object of type glmbenvelope.
family	family used for simulation. Used that this is different from the family used in other functions.
link	link function used for simulation.
progbar	dummy for flagging if a progressbar should be produced during the call

Details

This function uses the information contained in the constructed envelope list in order to sample from a model in standard form. The simulation proceeds as follows in order to generate each draw in the required number of samples.

1) A random number between 0 and 1 is generated and is used together with the information in the PLSD vector (from the envelope) in order to identify the part of the grid from which a candidate is to be generated.

2) For the part of the grid selected, the dimensions are looped through and a candidate component for each dimension is generated from a restricted normal using information from the Envelope (in particular, the values for logrt, loglt, and cbars corresponding to that the part of the grid selected and the dimension sampled)

3) The log-likelihood for the standardized model is evaluated for the generated candidate (note that the log-likelihood here includes the portion of the prior that was shifted to the log-likelihood)

4) An additional random number is generated and the log of this random number is compared to a log-acceptance rate that is calculated based on the candidate and the LLconst component from the Envelope component selected in order to determine if the candidate should be accepted or rejected

5) If the candidate was not accepted, the process above is repeated from step 1 until a candidate is accepted

Value

A list consisting of the following:

out	A matrix with simulated draws from a model in standard form. Each row represents one draw from the density
draws	A vector with the number of candidates required before acceptance for each draw

Examples

```
data(menarche2)

summary(menarche2)
plot(Menarche/Total ~ Age, data=menarche2)

Age2=menarche2$Age-13

x<-matrix(as.numeric(1.0),nrow=length(Age2),ncol=2)
x[,2]=Age2

y=menarche2$Menarche/menarche2$Total
wt=menarche2$Total

mu<-matrix(as.numeric(0.0),nrow=2,ncol=1)
mu[2,1]=(log(0.9/0.1)-log(0.5/0.5))/3

V1<-1*diag(as.numeric(2.0))

# 2 standard deviations for prior estimate at age 13 between 0.1 and 0.9
## Specifies uncertainty around the point estimates

V1[1,1]<-((log(0.9/0.1)-log(0.5/0.5))/2)^2
V1[2,2]=(3*mu[2,1]/2)^2  # Allows slope to be up to 1 times as large as point estimate

#out<-rglmb(n = 1000, y=y, x=x, mu=mu, P=solve(V1), wt = wt,
#           family = binomial(logit), Gridtype = 3)
#summary(out)

famfunc<-glmbfamfunc(binomial(logit))

f1<-famfunc$f1
f2<-famfunc$f2  # Used in optim and glmbsim_cpp
f3<-famfunc$f3  # Used in optim
f5<-famfunc$f5
f6<-famfunc$f6

dispersion2<-as.numeric(1.0)
start <- mu
offset2=rep(as.numeric(0.0),length(y))
P=solve(V1)
n=1000


###### Adjust weight for dispersion

wt2=wt/dispersion2
```

```
######################### Shift mean vector to offset so that adjusted model has 0 mean

alpha=x%*%as.vector(mu)+offset2
mu2=0*as.vector(mu)
P2=P
x2=x


#####  Optimization step to find posterior mode and associated Precision

parin=start-mu

opt_out=optim(parin,f2,f3,y=as.vector(y),x=as.matrix(x),mu=as.vector(mu2),
              P=as.matrix(P),alpha=as.vector(alpha),wt=as.vector(wt2),
              method="BFGS",hessian=TRUE
)

bstar=opt_out$par  ## Posterior mode for adjusted model
bstar
bstar+as.vector(mu)  # mode for actual model
A1=opt_out$hessian # Approximate Precision at mode

## Standardize Model

Standard_Mod=glmb_Standardize_Model(y=as.vector(y), x=as.matrix(x),P=as.matrix(P),
                                    bstar=as.matrix(bstar,ncol=1), A1=as.matrix(A1))

bstar2=Standard_Mod$bstar2
A=Standard_Mod$A
x2=Standard_Mod$x2
mu2=Standard_Mod$mu2
P2=Standard_Mod$P2
L2Inv=Standard_Mod$L2Inv
L3Inv=Standard_Mod$L3Inv

Env2=EnvelopeBuild(as.vector(bstar2), as.matrix(A),y, as.matrix(x2),
                   as.matrix(mu2,ncol=1),as.matrix(P2),as.vector(alpha),as.vector(wt2),
                   family="binomial",link="logit",Gridtype=as.integer(3),
                   n=as.integer(n),sortgrid=TRUE)

## These now seem to match

Env2

#int n, NumericVector y, NumericMatrix x, NumericMatrix mu, NumericMatrix P,
#NumericVector alpha, NumericVector wt, Function f2,
#Rcpp::List Envelope, Rcpp::CharacterVector family,
#Rcpp::CharacterVector link, int progbar

### Note: getting the types correct here is important but potentially difficult for users
### May be better to call an R function wrapper that checks (and converts when possible)
##  to correct types

sim=rnnorm_reg_std(n=as.integer(n),y=as.vector(y),x=as.matrix(x2),mu=as.matrix(mu2,ncol=1),
                   P=as.matrix(P2),alpha=as.vector(alpha),wt=as.vector(wt2),
                   f2=f2,Envelope=Env2,family="binomial",link="logit",as.integer(0))
```

```
out=L2Inv%*%L3Inv%*%t(sim$out)

for(i in 1:n){
  out[,i]=out[,i]+mu
}

summary(t(out))
mean(sim$draws)
```

`rnnorm_reg_std_cpp_parallel` *Parallel Truncated-Normal Regression (Standard Worker)*

Description

Wraps the C++ function ‘rnnorm_reg_std_cpp_parallel‘ for fast, threaded sampling.

Usage

```
rnnorm_reg_std_cpp_parallel(
  n,
  y,
  x,
  mu,
  P,
  alpha,
  wt,
  f2,
  Envelope,
  family,
  link,
  progbar = 1L
)
```

Arguments

`n`	integer; number of observations.
`y`	numeric vector of length n; response values.
`x`	numeric matrix with n rows; predictors.
`mu`	numeric matrix (p × . . .); prior means for each coefficient.
`P`	numeric matrix or block-matrix; prior precision/covariance.
`alpha`	numeric vector of length p; regression coefficients.
`wt`	numeric vector of length n; observation weights.
`f2`	R function; auxiliary update callback used internally.
`Envelope`	list with the following elements: - PLSD: numeric vector of slice-sampling probabilities. - loglt: numeric matrix of lower-tail log-densities. - logrt: numeric matrix of upper-tail log-densities. - cbars: numeric matrix of truncation bounds. - LLconst: numeric vector of log-likelihood offsets.

family	character string; distribution family (e.g. "gaussian").
link	character string; link function (e.g. "identity").
progbar	integer (0 or 1); whether to display a progress bar.

Value

A list with components: - out: numeric matrix (n × p) of sampled latent values. - draws: numeric vector of length n of final draws.

---

rNormal_Gamma_reg *The Bayesian Normal-Gamma Regression Distribution*

Description

rNormal_Gamma_reg is used to generate iid samples from Bayesian linear models with a normal-gamma prior. The model is specified by providing a data vector, a design matrix, and 4 prior constants.

Usage

```
rNormal_Gamma_reg(
  n,
  y,
  x,
  prior_list,
  offset = NULL,
  weights = 1,
  family = gaussian()
)
```

Arguments

n	number of draws to generate. If length(n) > 1, the length is taken to be the number required.
y	a vector of observations of length m.
x	a design matrix of dimension m * p.
prior_list	a list with the prior parameters (mu, Sigma, shape and rate) for the prior distribution.
offset	this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
family	a description of the error distribution and link function to be used in the model. For glm this can be a character string naming a family function, a family function or the result of a call to a family function. For glm.fit only the third option is supported. (See family for details of family functions.)

Details

The rNormal_Gamma_reg function produces iid samples for Bayesian generalized linear models from the gaussian family (identity link) with a conjugate multivariate normal-gamma prior for the regression coefficients and the dispersion (variance). See Raiffa and Schlaifer (1961) for details on conjugate priors.

Core required inputs for the function include the data vector, the design matrix and a prior specification. The function returns the simulated Bayesian coefficients and some associated outputs. The iid samples from the posterior density is genererated using standard simulation procedures for multivariate normal and gamma distributions.

Value

rNormal_Gamma_reg returns a object of class "rglmb". The function summary (i.e., summary.rglmb) can be used to obtain or print a summary of the results. The generic accessor functions coefficients, fitted.values, residuals, and extractAIC can be used to extract various useful features of the value returned by rNormal_Gamma_reg. An object of class "rglmb" is a list containing at least the following components:

coefficients	a n by length(mu) matrix with one sample in each row
PostMode	a vector of length(mu) with the estimated posterior mode coefficients
Prior	A list with two components. The first being the prior mean vector and the second the prior precision matrix
iters	an n by 1 matrix giving the number of candidates generated before acceptance for each sample.
famfunc	an object of class "famfunc"
Envelope	an object of class "envelope"
dispersion	an n by 1 matrix with simulated values for the dispersion
loglike	a n by 1 matrix containing the negative loglikelihood for each sample.

References

Dobson, A. J. (1990) *An Introduction to Generalized Linear Models.* London: Chapman and Hall.

Hastie, T. J. and Pregibon, D. (1992) *Generalized linear models.* Chapter 6 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole. McCullagh P. and Nelder, J. A. (1989) *Generalized Linear Models.* London: Chapman and Hall.

Nygren, K.N. and Nygren, L.M (2006) Likelihood Subgradient Densities. *Journal of the American Statistical Association*. vol.101, no.475, pp 1144-1156.

Raiffa, Howard and Schlaifer, R (1961) *Applied Statistical Decision Theory.* Boston: Clinton Press, Inc.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* New York: Springer.

See Also

pfamily for the list of available pfamilies and how they are specified. Each pfamily has a specific assigned simulation function. This particular simulation function is the simulation function for the dNormal_Gamma pfamily.

rglmb and glmb for functions that depend on simulation functions.

Other family simulation functions: rGamma_reg(), rNormal_reg(), rindependent_norm_gamma_reg()

Examples

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

ps=Prior_Setup(weight ~ group)
mu=ps$mu
V=ps$Sigma
mu[1,1]=mean(weight)

Prior_Check(weight ~ group,family =gaussian(),
           pfamily=dNormal(mu=mu,Sigma=V))

## Will move this step inside the Prior_Check
lm.D9 <- lm(weight ~ group,x=TRUE,y=TRUE)
disp_ML=sigma(lm.D9)^2
n_prior=2
shape=n_prior/2
rate= disp_ML*shape

lmb.D9=lmb(weight ~ group,dNormal_Gamma(mu,V/disp_ML,shape=shape,rate=rate))
summary(lmb.D9)

n<-10000
y<-lmb.D9$y
x<-as.matrix(lmb.D9$x)
prior_list=list(mu=mu,Sigma=V/disp_ML,shape=shape,rate=rate)
ngamma.D9=rNormal_Gamma_reg(n=1000,y=y,x=x,
  prior_list=prior_list)
summary(ngamma.D9$coefficients)
```

`rNormal_reg` *The Bayesian Generalized Linear Model with Normal Prior Distribution*

Description

`rNormal_reg` is used to generate iid samples from Bayesian Generalized linear models with a normal prior. The model is specified by providing a data vector, a design matrix, and 2 prior constants (mu and Sigma) for the normal prior.

Usage

```
rNormal_reg(
  n,
  y,
  x,
  prior_list,
  offset = NULL,
```

```
  weights = 1,
  family = gaussian()
)
```

Arguments

n	number of draws to generate. If length(n) > 1, the length is taken to be the number required.
y	a vector of observations of length m.
x	a design matrix of dimension m * p.
prior_list	a list with the two parameters (mu and Sigma) for the prior distribution.
offset	this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
family	a description of the error distribution and link function to be used in the model. For glm this can be a character string naming a family function, a family function or the result of a call to a family function. For glm.fit only the third option is supported. (See family for details of family functions.)

Details

The rNormal_reg function produces iid samples for Bayesian generalized linear models. with multivariate-normal priors. Core required inputs for the function include the data vector, the design matrix and a prior specification (provided in a list when this function is called directly). Specifying the pfamily as dNormal in the lmb or glmb is equivalent to calling this function directly.

The function returns the simulated Bayesian coefficients and some associated outputs. While it is possible to call this function directly, it is generally recommended that the lmb, rlmb, glmb or rglmb functions be used instead as those functions have more documentation and the resulting objects come with more methods.

When the specified family is gaussian, the multivariate normal is the conjugate prior distribution for the likelihood function and the posterior distribution is therefore also a multivariate normal density. Fairly standard procedures are applied in order to generate samples in that case. Currently, this includes using a Cholesky decomposition. Later implementations may switch to a QR decomposition to increase consistency with the lm function and to increase associated numerical accuracy.

When the specified family is any log-concave non-gaussian family, then the estimation uses the Likelihood subgradient density approach of Nygren and Nygren (2006). This approaches uses tangencies to the log-likelihood function in order to construct an enveloping function from which candidate draws are generated and then either accepted/rejected using accept/reject methods. The core C function performing this simulation essentially goes through the following steps:

1) The model is standardized to have prior mean vector equal to 0 (i.e., offsets and any prior mean are combined into a constant term).

2) The posterior mode for this transformed model is found. Currently this uses the optim function. Later implementations may replace this with iteratively reweighted least squares (IWLS) to increase consistency with the glm function and to enhance numerical accuracy.

3) The model is further standardized so that (a) the precision matrix at the posterior mode is diagonal and (b) the prior variance-covariance matrix is the identity matrix (see glmb_Standardize_Model for details).

4) An enveloping function is built for the the standardized model, containing constants needed during simulation (see EnvelopeBuild for details).

5) Samples for the standardized model are generated using accept-reject methods (see rnnorm_reg_std for details).

6) The output from the standardized model are transformed back to the original scale by reversing the two eigenvalue decompositions and by adding back the prior mean.

Value

rNormal_reg returns a object of class "rglmb". The function summary (i.e., summary.rglmb) can be used to obtain or print a summary of the results. The generic accessor functions coefficients, fitted.values, residuals, and extractAIC can be used to extract various useful features of the value returned by rNormal_reg. An object of class "rglmb" is a list containing at least the following components:

coefficients	a n by length(mu) matrix with one sample in each row
PostMode	a vector of length(mu) with the estimated posterior mode coefficients
Prior	A list with two components. The first being the prior mean vector and the second the prior precision matrix
iters	an n by 1 matrix giving the number of candidates generated before acceptance for each sample.
famfunc	an object of class "famfunc"
Envelope	an object of class "envelope"
dispersion	an n by 1 matrix with simulated values for the dispersion
loglike	a n by 1 matrix containing the negative loglikelihood for each sample.

References

Dobson, A. J. (1990) *An Introduction to Generalized Linear Models.* London: Chapman and Hall.

Hastie, T. J. and Pregibon, D. (1992) *Generalized linear models.* Chapter 6 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole. McCullagh P. and Nelder, J. A. (1989) *Generalized Linear Models.* London: Chapman and Hall.

Nygren, K.N. and Nygren, L.M (2006) Likelihood Subgradient Densities. *Journal of the American Statistical Association*. vol.101, no.475, pp 1144-1156.

Raiffa, Howard and Schlaifer, R (1961) *Applied Statistical Decision Theory.* Boston: Clinton Press, Inc.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* New York: Springer.

See Also

Other family simulation functions: rGamma_reg(), rNormal_Gamma_reg(), rindependent_norm_gamma_reg()

Examples

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

ps=Prior_Setup(weight ~ group)
mu=ps$mu
V=ps$Sigma
mu[1,1]=mean(weight)

Prior_Check(weight ~ group,family =gaussian(),
           pfamily=dNormal(mu=mu,Sigma=V))

## Will move this step inside the Prior_Check
lm.D9 <- lm(weight ~ group,x=TRUE,y=TRUE)
disp_ML=sigma(lm.D9)^2
n_prior=2
shape=n_prior/2
rate= disp_ML*shape

lmb.D9=lmb(weight ~ group,dNormal_Gamma(mu,V/disp_ML,shape=shape,rate=rate))
summary(lmb.D9)

n<-10000
y<-lmb.D9$y
x<-as.matrix(lmb.D9$x)
prior_list=list(mu=mu,Sigma=V/disp_ML,shape=shape,rate=rate)
ngamma.D9=rNormal_Gamma_reg(n=1000,y=y,x=x,
  prior_list=prior_list)
summary(ngamma.D9$coefficients)
```

`rNormal_reg.wfit` *Fitter Function for Bayesian Linear Models*

Description

Basic computing engine called to find the posterior mode and a UL decomposition

Usage

```
rNormal_reg.wfit(
  x,
  y,
  P,
  mu,
  w,
  offset = NULL,
  method = "qr",
  tol = 1e-07,
```

```
  singular.ok = TRUE,
  ...
)
```

Arguments

x	design matrix of dimension n * p.
y	vector of observations of length n, or a matrix with n rows.
P	Prior precision matrix of dimension p * p.
mu	Prior mean vector of length p.
w	vector of weights (length n) to be used in the fitting process for the wfit functions. Weighted least squares is used with weights w, i.e., sum(w * e^2) is minimized.
offset	(numeric of length n). This can be used to specify an *a priori* known component to be included in the linear predictor during fitting.
method	currently, only method = "qr" is supported.
tol	tolerance for the qr decomposition. Default is 1e-7.
singular.ok	logical. If FALSE, a singular model is an error.
...	currently disregarded.

Value

a list wih components:

Examples

```
## ----dobson-------------------------------------------------------------------
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)

## Prior mean vector
mu<-matrix(0,5)
mu[1,1]=log(mean(counts))
## Prior standard deviation and Variance
mysd<-1
V=((mysd)^2)*diag(5)
## Call to glmb
glmb.D93<-glmb(n=1000,counts ~ outcome + treatment,
               family = poisson(),pfamily=dNormal(mu=mu,Sigma=V))
## ----glmb confint-------------------------------------------------------------
confint(glmb.D93)
```

setlogP *Calculate constants used during sampling from Likelihood subgradient densities*

Description

This function computes two vectors of constants needed during sampling. The first is used to determine the probability with which each component of the grid should be visited, while the second is used as a constants when calculating acceptance rates.

Usage

```
setlogP(logP, NegLL, cbars, G3)
```

Arguments

logP	A matrix that typically contains two columns and information for each component of the grid. The first column will typically hold the final output from the Set_Grid function, which is the density associated with the related restricted normal.
NegLL	A vector with evaluations of the Negative Log-likelihood for each of the components of the grid.
cbars	A matix holding the gradients for the Negative of the Log-likelihood for each of the componentsof the grid.
G3	A matrix containing the set of tangent points used in the grid.

Value

Refer to Nygren and Nygren (2006) for details. The first .

logP	The first column holds the value passed into the function while the second contains the log of the (un-normalized) probabilities with which each of the components of the grid should be visited. This corresponds to the log of the denominator components used to compute p_i in remark 6 in Nygren and Nygren (2006)
LLconst	This holds a vector of constants used as upper bounds when deriving acceptance rates during the accept-reject sampling process. This constant corresponds to the denominator for the function h() in Theorem 1 of Nygren and Nygren (2006). During the sampling, the log of the numerator of the same function if evaluated for each candidate and the difference between the candidate value and this constant is used to determine the acceptance rate to use when evaluating acceptance of the candidate. If the evaluated value for the candidate is close to this constant, then the chance of acceptance rate is high. If it is much smaller, then the chance of acceptance is low.

Examples

```
## ----dobson-------------------------------------------------------------------
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
```

```
treatment <- gl(3,3)

## Prior mean vector
mu<-matrix(0,5)
mu[1,1]=log(mean(counts))
## Prior standard deviation and Variance
mysd<-1
V=((mysd)^2)*diag(5)
## Call to glmb
glmb.D93<-glmb(n=1000,counts ~ outcome + treatment,
               family = poisson(),pfamily=dNormal(mu=mu,Sigma=V))
## ----glmb extractAIC----------------------------------------------------------
extractAIC(glmb.D93)
```

`Set_Grid` *Calculate Log-densities for Grid Components*

Description

Computes un-normalized log of the density associated with each component of a Grid used to sample using the likelihood-subgradient density approach

Usage

```
Set_Grid(GridIndex, cbars, Lint)
```

Arguments

`GridIndex`	A matrix containing information for each component in the grid related to whether the components is to the left, in the center, or to the right of the density. Each row corresponds to a component in the grid, while the columns correspond to the transformed (standardized) variables.
`cbars`	A matrix containing the subgradient for the (adjusted) negative log-likelihood function for each the component in the grid.
`Lint`	A matrix storing information on where the upper and lower bounds are, depending on whether the sampling is from the left, the center, or the right.

Value

Refer to Nygren and Nygren (2006) for details. The first set of items refers to Example 2 in section 3.1. All except the last item in this list of returned items has a number of rows equaling the number of components of the grid and a number of columns equaling the number of coefficients in the model. All quantities refer to the respective coefficient for each of the components of the grid.

`Down`	The lower bounds for the interval to be evaluated. Either negative infinity or a real number.
`Up`	The upper bounds for the interval to be evaluated. Either positive infinity or a real number.
`lglt`	The log of the density between negative infinity and the upper bound
`lgrt`	The log of the density between the lower bound and infinity
`lgct`	The log of the density between the lower and upper bounds

logU	The one of the 3 above that is relevant for the component of the grid
logP	A two column matrix, the first of which holds sum of logU across the components. The second column is 0 and is later populated by the Set_logP function

Examples

```
## ----dobson----------------------------------------------------------------
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)

## Prior mean vector
mu<-matrix(0,5)
mu[1,1]=log(mean(counts))
## Prior standard deviation and Variance
mysd<-1
V=((mysd)^2)*diag(5)
## Call to glmb
glmb.D93<-glmb(n=1000,counts ~ outcome + treatment,
               family = poisson(),pfamily=dNormal(mu=mu,Sigma=V))
## ----glmb extractAIC-------------------------------------------------------
extractAIC(glmb.D93)
```

---

simulate.glmb *Simulate Responses*

Description

Simulate responses from the distribution corresponding to a fitted glmb object.

Usage

```
## S3 method for class 'glmb'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

object	An object of class glmb, typically the result of a call to the function glmb.
nsim	Defunct (see below).
seed	an object specifying if and how the random number generator should be initialized (seeded).
...	Additional arguments passed to the function. Will frequently include a matrix pred of simulated predictions from the predict function, the family (e.g., binomial) and an optional vector of weights specifying prior.weights for the simulated values (default is 1)

Value

Simulated values for data corresponding to simulated model predictions that correspond either to the original data or to a newdata data frame provided to the predict function.

Examples

```
data(menarche2)
## ----Analysis Setup--------------------------------------------------------
## Number of variables in model
Age=menarche2$Age
nvars=2
## Reference Ages for setting of priors and Age_Difference
ref_age1=13  # user can modify this
ref_age2=15  ## user can modify this
## Define variables used later in analysis
Age2=menarche2$Age-ref_age1
Age_Diff=ref_age2-ref_age1
mu1=as.matrix(c(0,1.098612),ncol=1)
V1<-1*diag(nvars)
V1[1,1]=0.18687882
V1[2,2]=0.10576217
V1[1,2]=-0.03389182
V1[2,1]=-0.03389182
Menarche_Model_Data=data.frame(Age=menarche2$Age,Total=menarche2$Total,
                               Menarche=menarche2$Menarche,Age2)
glmb.out1<-glmb(n=1000,cbind(Menarche, Total-Menarche) ~Age2,family=binomial(logit),
                pfamily=dNormal(mu=mu1,Sigma=V1),data=Menarche_Model_Data)

# Prediction from original model
pred1=predict(glmb.out1,type="response")

## Get Original Residuals, their means, and credible bounds
res_out=residuals(glmb.out1)
colMeans(res_out)

## Set up to simulate new data and residuals
res_mean=colMeans(res_out)
res_low1=apply(res_out,2,FUN=quantile,probs=c(0.025))
res_high1=apply(res_out,2,FUN=quantile,probs=c(0.975))

## Simulate new data and get residuals for simulated data

ysim1=simulate(glmb.out1,nsim=1,seed=NULL,pred=pred1,family="binomial",
               prior.weights=weights(glmb.out1))


res_ysim_out1=residuals(glmb.out1,ysim=ysim1)
res_low=apply(res_ysim_out1,2,FUN=quantile,probs=c(0.025))
res_high=apply(res_ysim_out1,2,FUN=quantile,probs=c(0.975))

# Plot Credible Interval bounds for Deviance Residuals

plot(res_mean~Age,ylim=c(-2.5,2.5),
main="Credible Interval Bound for Menarche - Logit Model Deviance Residuals",
xlab = "Age", ylab = "Avg. Dev. Res")
lines(Age, 0*res_mean,lty=1)
lines(Age, res_low,lty=1)
lines(Age, res_high,lty=1)
lines(Age, res_low1,lty=2)
lines(Age, res_high1,lty=2)
```

summary.glmb *Summarizing Bayesian Generalized Linear Model Fits*

Description

These functions are all methods for class glmb or summary.glmb objects.

Usage

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

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

Arguments

object	an object of class "glmb" for which a summary is desired.
x	an object of class "summary.glmb" for which a printed output is desired.
digits	the number of significant digits to use when printing.
...	Additional optional arguments

Details

The summary.glmb function summarizes the output from the glmb function. Key output includes mean residuals, information related to the prior, mean coefficients with associated stats, percentiles for the coefficients, as well as the effective number of parameters and the DIC statistic.

Value

summary.glmb returns a object of class "summary.glmb", a list with components:

call	the component from object
n	number of draws generated
residuals	vector of mean deviance residuals
coefficients1	Matrix with the prior mean and maximum likelihood coefficients with associated standard deviations
coefficients	Matrix with columns for the posterior mode, posterior mean, posterior standard deviation, monte carlo error, and tail probabilities (posterior probability of observing a value for the coefficient as extreme as the prior mean)
Percentiles	Matrix with estimated percentiles associated with the posterior density
pD	Estimated effective number of parameters
deviance	Vector with draws for the deviance
DIC	Estimated DIC statistic
iters	Average number of candidates per generated draws

See Also

lmb, glmb, summary, [stats]summary.lm,[stats]summary.glm.

Examples

```
##########################  Example for glmb function:
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
print(d.AD <- data.frame(treatment, outcome, counts))

mysd<-1
mu<-matrix(0,5)
mu[1,1]=log(mean(counts))
V0<-((mysd)^2)*diag(5)
glmb.D93<-glmb(counts ~ outcome + treatment, family = poisson(),
pfamily=dNormal(mu=mu,Sigma=V0))
summary(glmb.D93)

##########################  Example for lmb function

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

ps=Prior_Setup(weight ~ group)
mu=ps$mu
V=ps$Sigma
mu[1,1]=mean(weight)

Prior_Check(weight ~ group,family =gaussian(),
            pfamily=dNormal(mu=mu,Sigma=V))

## May move this step inside the Prior_Check function
lm.D9 <- lm(weight ~ group,x=TRUE,y=TRUE)
disp_ML=sigma(lm.D9)^2
n_prior=2
shape=n_prior/2
rate= disp_ML*shape

lmb.D9=lmb(weight ~ group,dNormal_Gamma(mu,V/disp_ML,shape=shape,rate=rate))
summary(lmb.D9)
```

`summary.rglmb` *Summarizing Bayesian Generalized Linear Model Distribution Functions*

Description

These functions are all `methods` for class `rglmb` or `summary.rglmb` objects.

Usage

```
## S3 method for class 'rglmb'
```

```
summary(object, ...)

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

Arguments

object	an object of class "rglmb" for which a summary is desired.
x	an object of class "summary.rglmb" for which a printed output is desired.
digits	the number of significant digits to use when printing.
...	Additional optional arguments

Details

The summary.rglmb function summarizes the output from the rglmb function. It takes an object of class rglmb as an input. The output to a large extent mirrors the output from the summary.glm function. This is particularly true for the print output from the function (i.e. the output of the function print.summary.rglmb).

Value

summary.rglmb returns a object of class "summary.rglmb", a list with components:

n	number of draws generated
coefficients1	Matrix with the prior mean and approximate weight for the prior relative to the data
coefficients	Matrix with columns for the posterior mode, posterior mean, posterior standard deviation, monte carlo error, and tail probabilities (posterior probability of observing a value for the coefficient as extreme as the prior mean)
Percentiles	Matrix with estimated percentiles associated with the posterior density

See Also

lmb, glmb, summary, [stats]summary.lm,[stats]summary.glm.

Examples

```
data(menarche2)

summary(menarche2)
plot(Menarche/Total ~ Age, data=menarche2)

Age2=menarche2$Age-13

x<-matrix(as.numeric(1.0),nrow=length(Age2),ncol=2)
x[,2]=Age2

y=menarche2$Menarche/menarche2$Total
wt=menarche2$Total

mu<-matrix(as.numeric(0.0),nrow=2,ncol=1)
mu[2,1]=(log(0.9/0.1)-log(0.5/0.5))/3
```

```
V1<-1*diag(as.numeric(2.0))

# 2 standard deviations for prior estimate at age 13 between 0.1 and 0.9
## Specifies uncertainty around the point estimates

V1[1,1]<-((log(0.9/0.1)-log(0.5/0.5))/2)^2
V1[2,2]=(3*mu[2,1]/2)^2  # Allows slope to be up to 3 times as large as point estimate

out<-rglmb(n = 1000, y=y, x=x, pfamily=dNormal(mu=mu,Sigma=V1), weights = wt,
           family = binomial(logit))
summary(out)

print(summary(out),digits=4)

mean(out$iters)
```

---

vcov.glmb *Calculate Variance-Covariance Matrux for a Fitted Model Object*

Description

Returns the variance-covariance matrix of the main parameters of a fitted model object.

Usage

```
## S3 method for class 'glmb'
vcov(object, ...)
```

Arguments

object	fitted model object, typically the result of a call to "glmb".
...	additional arguments for method functions.

Value

A matrix of estimated covariances between the parameter estimates in the linear or non-linear predictor of the model. This should have row and column names corresponding to the parameter names given by the coef method.

Examples

```
## ----dobson-------------------------------------------------------------------
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)

## Prior mean vector
mu<-matrix(0,5)
mu[1,1]=log(mean(counts))
## Prior standard deviation and Variance
mysd<-1
V=((mysd)^2)*diag(5)
```

```
## Call to glmb
glmb.D93<-glmb(n=1000,counts ~ outcome + treatment,
               family = poisson(),pfamily=dNormal(mu=mu,Sigma=V))

## ----glmb vcov-----------------------------------------------------------------
vcov(glmb.D93)
```

Index

∗ **datasets, Bayesian Binomial Regression**
AMI, 3
menarche2, 50
∗ **datasets, Bayesian Poisson Regression, Bayesian Gamma Regression**
carinsca, 5
∗ **internal**
EnvelopeBuild, 10
EnvelopeOpt, 13
EnvelopeSort, 15
glmb_Standardize_Model, 38
glmbayes, 34
glmbayes-package, 2
glmbfamfunc, 37
Neg_logLik, 51
Normal_ct, 51
rnnorm_reg_std, 79
Set_Grid, 92
setlogP, 91
∗ **modelfuns**
glmb, 23
lmb, 45
rglmb, 66
rlmb, 76
∗ **prior**
Prior_Check, 57
Prior_Setup, 60
∗ **simfuncs**
rGamma_reg, 63
rindependent_norm_gamma_reg, 69
rNormal_Gamma_reg, 84
rNormal_reg, 86
[.data.frame, *60*

AMI, 3
anova.glmb, 3
as.data.frame, *24*, *45*, *58*, *60*

binomial, *26*, *48*

carinsca, 5
case.names.glmb, 6
coef, *98*
coefficients, *25*, *46*, *64*, *67*, *71*, *77*, *85*, *88*
confint.glmb, 7
cooks.distance.glmb *(*glmb.influence.measures*)*, 28

deviance.rglmb, 8
dfbetas.glmb *(*glmb.influence.measures*)*, 28
dGamma *(*pfamily*)*, 52
dIndependent_Normal_Gamma *(*pfamily*)*, 52
dNormal, *87*
dNormal *(*pfamily*)*, 52
dNormal_Gamma, *85*
dNormal_Gamma *(*pfamily*)*, 52
dummy.coef.glmb, 9, *27*, *48*, *68*, *78*

EnvelopeBuild, 10, *14*, *25*, *46*, *77*, *88*
EnvelopeOpt, 13
EnvelopeSort, *14*, 15
extractAIC, *25*, *47*, *64*, *67*, *71*, *77*, *85*, *88*
extractAIC.glmb, 19, *27*, *48*, *68*, *78*
extractAIC.rglmb, 20
extractDIC *(*extractAIC.glmb*)*, 19

f2_gaussian_vector, 21
family, *24*, *25*, *31*, *37*, *38*, *58*, *63*, *67*, *68*, *70*, *74*, *84*, *87*
fitted.values, *25*, *46*, *64*, *67*, *71*, *77*, *85*, *88*
formula, *23*, *45*, *58*, *60*
formula.summary.rglmb, 22

glm, *25–28*, *34*, *41*, *46*, *48*, *68*, *78*, *87*
glm.control, *24*, *58*
glmb, *4*, *7*, *19*, *22*, 23, *25*, *34*, *37*, *47–49*, *52*, *54*, *61*, *68*, *78*, *85*, *87*, *95*, *97*
glmb.covratio *(*glmb.influence.measures*)*, 28
glmb.dffits *(*glmb.influence.measures*)*, 28
glmb.influence.measures, 28
glmb.wfit, 31
glmb_Standardize_Model, 38, *88*
glmbayes, 34
glmbayes-package, 2
glmbfamfunc, 37

influence.glmb, 41
Inv_f3_gaussian, 43

list, *28*, *31*, *41*, *90*
lm, *27*, *28*, *41*, *46–48*, *68*, *78*, *87*
lmb, *27*, 45, *52*, *54*, *68*, *77*, *78*, *87*, *95*, *97*
logLik.glmb, 49

menarche2, 50
methods, *61*, *95*, *96*
model.extract, *24*, *58*, *67*
model.offset, *63*, *80*, *84*, *87*

na.action, *60*
na.fail, *24*, *46*, *58*
napredict, *56*
Neg_logLik, 51
Neg_logLik2 *(*Neg_logLik*)*, 51
Normal_ct, 51

offset, *24*, *58*, *67*
optim, *87*
options, *24*, *46*, *58*, *60*

PackageOverview *(*glmbayes*)*, 34
pfamily, *24*, *27*, *45*, *46*, *48*, 52, *58*, *67*, *68*, *76*, *78*, *85*, *87*
pnorm_ct *(*Normal_ct*)*, 51
predict.glmb, *27*, *48*, 55, *68*, *78*
print.dummy_coef.glmb *(*dummy.coef.glmb*)*, 9
print.glmb *(*glmb*)*, 23
print.glmbfamfunc *(*glmbfamfunc*)*, 37
print.lmb *(*lmb*)*, 45
print.pfamily *(*pfamily*)*, 52
print.rGamma_reg *(*rGamma_reg*)*, 63
print.rglmb *(*rglmb*)*, 66
print.rlmb *(*rlmb*)*, 76
print.summary.glmb *(*summary.glmb*)*, 95
print.summary.rglmb, *97*
print.summary.rglmb *(*summary.rglmb*)*, 96
Prior_Check, *27*, *48*, 57, *61*, *68*, *78*
Prior_Setup, *27*, *48*, *59*, 60, *68*, *78*

qr, *31*, *90*

residuals, *25*, *47*, *64*, *67*, *71*, *77*, *85*, *88*
residuals.glmb, *27*, 61, *68*
residuals.lmb *(*residuals.glmb*)*, 61
rGamma_reg, *34*, *54*, 63, *64*, *71*, *85*, *88*
rglmb, *8*, *14*, *20*, *27*, *34*, *37*, *48*, *52*, *54*, *64*, 66, *67*, *77*, *78*, *85*, *87*, *97*
rindep_norm_gamma_reg_std_R, 73
rindependent_norm_gamma_reg, *64*, 69, *85*, *88*
rindependent_norm_gamma_reg_v2 *(*rindependent_norm_gamma_reg*)*, 69
rindependent_norm_gamma_reg_v3 *(*rindependent_norm_gamma_reg*)*, 69
rindependent_norm_gamma_reg_v4 *(*rindependent_norm_gamma_reg*)*, 69
rlmb, *27*, *48*, *54*, *68*, 76, *87*
rnnorm_reg_std, 79, *88*
rnnorm_reg_std_cpp_parallel, 83
rnorm, *67*
rnorm_ct *(*Normal_ct*)*, 51
rNormal_Gamma_reg, *54*, *64*, *71*, 84, *85*, *88*
rNormal_reg, *54*, *64*, *71*, *85*, 86, *88*
rNormal_reg.wfit, 89
rstandard.glmb *(*glmb.influence.measures*)*, 28
rstudent.glmb *(*glmb.influence.measures*)*, 28

Set_Grid, *11*, 92
setlogP, 91
simulate.glmb, *27*, *48*, *68*, *78*, 93
summary, *95*, *97*
summary.glm, *95*, *97*
summary.glmb, *25*, *27*, *46*, *48*, *68*, *78*, 95
summary.lm, *95*, *97*
summary.rGamma_reg *(*rGamma_reg*)*, 63
summary.rglmb, *64*, *67*, *71*, *77*, *85*, *88*, 96

terms, *26*, *47*, *60*

variable.names.glmb *(*case.names.glmb*)*, 6
vcov.glmb, 98