Package ‘DPpackage’

January 6, 2018

Version 1.1-7.4

Date 2018-01-05

Title Bayesian Nonparametric Modeling in R

Depends R (>= 2.10)

Imports MASS, nlme, survival, splines, methods

Description Functions to perform inference via
simulation from the posterior distributions for Bayesian
nonparametric and semiparametric models. Although the name of
the package was motivated by the Dirichlet Process prior, the
package considers and will consider other priors on functional
spaces. So far, DPpackage includes models considering Dirichlet
Processes, Dependent Dirichlet Processes, Dependent Poisson-
Dirichlet Processes, Hierarchical Dirichlet Processes, Polya
Trees, Linear Dependent Tailfree Processes, Mixtures of
Triangular distributions, Random Bernstein polynomials priors
and Dependent Bernstein Polynomials. The package also includes
models considering Penalized B-Splines.
Includes semiparametric models for marginal and conditional
density estimation, ROC curve analysis, interval censored data,
binary regression models, generalized linear mixed models, IRT
type models, and generalized additive models. Also
contains functions to compute Pseudo-Bayes factors for model
comparison, and to elicitate the precision parameter of the
Dirichlet Process. To maximize computational efficiency, the
actual sampling for each model is done in compiled FORTRAN. The
functions return objects which can be subsequently analyzed
with functions provided in the 'coda' package.

License GPL (>= 2)

URL http://www.mat.puc.cl/~ajara

Author Alejandro Jara [aut, cre], Timothy Hanson [ctb], Fernando
Quintana [ctb], Peter Mueller [ctb], Gary Rosner [ctb]

Maintainer ORPHANED

NeedsCompilation yes

X-CRAN-Original-Maintainer Alejandro Jara <atjara@uc.cl>

X-CRAN-Comment Orphaned and corrected on 2017-09-15 as errors were not corrected despite repeated reminders.

Repository CRAN

Date/Publication 2018-01-06 08:39:08 UTC

R topics documented:

BDPdensity 4
bir 7
calgb 8
calgb.pred 9
CSDPbinary 10
deterioration 14
DPbetabinom 15
DPbinary 18
DPcaterpillar 22
DPcdensity 23
DPdensity 28
DPelicit 34
DPglmm 36
DPlmm 40
DPMdencens 44
DPmeta 49
DPMglmm 53
DPMlmm 57
DPMmeta 61
DPMolmm 65
DPMrandom 69
DPMrasch 71
DPMraschpoisson 76
DPmultmeta 80
DPolmm 84
DPpsBF 88
DPrandom 89
DPrasch 91
DPraschpoisson 95
DProc 100
DPsurvint 104
fleabeetles 109
FPTbinary 110
FPTrasch 113
FPTraschpoisson 118
fractionation 122
galaxy 123
HDPMcdensity 124
HDPMdensity 128

hiv 132
igg 133
indon 134
LDBDPdensity 135
LDDPdensity 140
LDDPrasch 145
LDDPraschpoisson 152
LDDProc 158
LDDPsurvival 165
LDDPtwopl 170
LDPDdoublyint 177
LDTFPdensity 182
LDTFPglmm 190
LDTFPsurvival 196
nodal 200
orings 202
Pbinary 203
Plm 206
predict.DPsurvint 208
predict.HDPMcdensity 211
predict.HDPMdensity 214
ps 216
PSgam 217
psychiatric 221
PTdensity 222
PTglmm 226
PTlm 231
PTlmm 235
PTmeta 240
PTolmm 244
PTrandom 248
PTsampler 250
rats 255
rolling 256
schoolgirls 257
seizures 258
sports 259
TDPdensity 260
toenail 263
uniond 264

Index **266**

`BDPdensity`	*Semiparametric Bayesian density estimation using Bernstein Polynomials*

Description

This function generates a posterior density sample for a Bernstein-Dirichlet model.

Usage

```
BDPdensity(y,support=3,ngrid=1000,grid=NULL,prior,mcmc,state,status,
          data=sys.frame(sys.parent()),na.action=na.fail)
```

Arguments

`y`	a vector giving the data from which the density estimate is to be computed.
`support`	an integer number giving the support of the random density, 1=[0,1], 2=(0, +Inf], and 3=(-In,+Inf). Depending on this, the data is transformed to lie in the [0,1] interval.
`ngrid`	number of grid points where the density estimate is evaluated. This is only used if dimension of `y` is lower or equal than 2. The default value is 1000.
`grid`	vector of grid points where the density estimate is evaluated. The default value is NULL and the grid is chosen according to the range of the data.
`prior`	a list giving the prior information. The list includes the following parameter: `aa0` and `ab0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `aa0` is missing, see details below), `a0` and `b0` giving the parameters of the beta centering distribution of the DP prior, and `kmax` giving the maximum value of the discrete uniform prior for the degree of the Bernstein polynomial.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `BDPdensity` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a Bernstein-Dirichlet model for density estimation (Petrone, 1999a, 1999b; Petrone and Waserman, 2002):

$$y_i|G \sim G, i = 1, \ldots, n$$

$$G|kmax, \alpha, G_0 \sim BDP(kmax, \alpha G_0)$$

where, y_i is the transformed data to lie in [0,1], `kmax` is the upper limit of the discrete uniform prior for the degree of the Bernstein polynomial, α is the total mass parameter of the Dirichlet process component, and G_0 is the centering distribution of the DP. The centering distribution corresponds to a $G_0 = Beta(a_0, b_0)$ distribution.

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

Value

An object of class `BDPdensity` representing the Bernstein-Dirichlet model fit. Generic functions such as `print`, `summary`, and `plot` have methods to show the results of the fit. The results include the degree of the polynomial `k`, `alpha`, and the number of clusters.

The MCMC samples of the parameters and the errors in the model are stored in the object `thetasave` and `randsave`, respectively. Both objects are included in the list `save.state` and are matrices which can be analyzed directly by functions provided by the coda package.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`ncluster`	an integer giving the number of clusters.
`yclus`	a real vector giving the `y` latent variables of the clusters (only the first `ncluster` are considered to start the chain).
`ss`	an interger vector defining to which of the `ncluster` clusters each observation belongs.
`alpha`	giving the value of the precision parameter.
`k`	giving the degree of the Bernstein polynomial.

Author(s)

Alejandro Jara <`<atjara@uc.cl>`>

Fernando Quintana <`<quintana@mat.puc.cl>`>

References

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Petrone, S. (1999a) Random Bernstein Polynomials. Scandinavian Journal of Statistics, 26: 373-393.

Petrone, S. (1999b) Bayesian density estimation using Bernstein polynomials. The Canadian Journal of Statistics, 27: 105-126.

Petrone, S. and Waserman, L. (2002) Consistency of Bernstein polynomial posterior. Journal of the Royal Statistical Society, Series B, 64: 79-100.

See Also

DPdensity, PTdensity

Examples

```
## Not run:
    # Data
      data(galaxy)
      galaxy<-data.frame(galaxy,speeds=galaxy$speed/1000)
      attach(galaxy)

    # Initial state
      state <- NULL

    # MCMC parameters

      nburn<-1000
      nsave<-10000
      nskip<-10
      ndisplay<-100
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay)

    # Prior
      prior<-list(aa0=2.01,
                  ab0=0.01,
                  kmax=1000,
                  a0=1,
                  b0=1)

    # Fitting the model

      fit <- BDPdensity(y=speeds,prior=prior,mcmc=mcmc,
                        state=state,status=TRUE)

      plot(fit)


## End(Not run)
```

`bir` *British Institute of Radiology Fractionation Studies*

Description

The British Institute of Radiology (BIR) conducted two large-scale randomized clinical trials to assess the effectiveness of different radiotherapy treatment schedules for cancer of the larynx and pharynx. The cambined data come from 858 subjects with laryngeal squamous cell carcinomas and no involvement of asjacent organs. These data have been described and analyzed by Rezvani, Fowler, Hopewell, and Alcock (1993).

Usage

```
data(bir)
```

Format

A data frame with 858 observations on the following 6 variables.

`response` Three-year local control 1 (0 no control).

`dose` Total dose (grays).

`df` Total dose x dose/fraction.

`time` Total time of treatment (days).

`kt2` Tumor status (indicators for 2nd level of factor).

`kt3` Tumor status (indicators for 3rd level of factor).

Details

Three-year local control - meaning no detection of laryngeal carcinoma within three years after treatment - is the binary response, coded as 1 if local control is achieved and 0 otherwise. For this data set, three-year local control is achieved for 69 a total dose of radiation is administered in fractions over a treatment period. The dose per fraction df is measured in grays (Gy), the length of treatment period time is measured in days, and the number of fractions of the dose is nf. Tumors are classified by stage (i.e., the extent of invasion), into three groups. This categorical covariate is coded by two indicator variables kt2 and kt3, which are defined by kt2=1 (kt3=1) is the tumor is stage II (stage III) and zero otherwise.

Chappell, Nondahl and Fowler (1995) argued that the tumor stage, the total dose, the total time, and the interaction of the total dose per fraction are the relevant explanantory variables affecting probability of local control.

References

Chappell, R., Nondahl, D.M., and Fowler, J.F. (1995) Modelling Dose and Local Control in Radiotheraphy. Journal of the American Statistical Asso- ciation, 90: 829 - 838.

Newton, M.A., Czado, C., and Chappell, R. (1996) Bayesian inference for semiparametric binary regression. Journal of the American Statistical Association, 91, 142-153.

Rezvani, M., Fowler, J., Hopewell, J., and Alcock, C. (1993) Sensitivity of Human Squamous Cell Carcinoma of the Larynx to Fractionated Radiotherapy. British Journal of Radiology, 66: 245 - 255.

Examples

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

---

calgb *Cancer and Leukemia Group B (CALGB) Data*

Description

Data from two studies carried out by the Cancer and Leukemia Group B (CLAGB): CALGB 8881 and CALGB 9160. In both studies, the main response was white blood cell count (WBC) for each patient over time. Mueller and Rosner (1998) used a non-linear patient specific regression model. The data consider the subject-specific regression parameters (Z1, Z2, Z3, T1, T2, B0, B1) and information on covariates.

CALGB has kindly agreed to make these data available for interested readers, subject to the following conditions: i) Any paper using these data should acknowledge CALGB for the use of the data, and ii) the paper should reference the original papers describing the studies.

Usage

```
data(calgb)
```

Format

A data frame with 98 observations on the following 12 variables.

`Z1` a numeric vector giving the estimated Z1 coefficients of the logistic regression curve.

`Z2` a numeric vector giving the estimated Z2 coefficients of the logistic regression curve.

`Z3` a numeric vector giving the estimated Z3 coefficients of the logistic regression curve.

`T1` a numeric vector giving the estimated time point where the horizontal line of the curve is defined, i.e., the curve consists of a horizontal line up to t=T1ji.

`T2` a numeric vector giving the estimated time point where the logistic component of the curve is defined, i.e., the curve consist of a logistic regression curve starting at t=T2ji.

`B0` a numeric vector giving the estimated B0 coefficients of the logistic regression curve.

`B1` a numeric vector giving the estimated B1 coefficients of the logistic regression curve.

`CTX` a numeric vector giving the dose level of cyclophosphamide.

`GM` a numeric vector giving the dose level GM-CSF.

`AMOF` a numeric vector giving the dose level of amifostine.

`pat` a numeric vector giving the patient indicators.

`study` a numeric vector giving the study indicators.

Source

CALGB 8881: Lichtman, S. M., Ratain, M. J., Echo, D. A., Rosner, G., Egorin, M. J., Budman, D. R., Vogelzang,N. J., Norton, L. and Schilsky, R. L. (1993) Phase I trial and granulocyte-macrophage colony-stimulating factor plus high-dose cyclophosphamide given every 2 weeks: a Cancer and Leukemia Group B study. Journal of the National Cancer Institute, 85: 1319-1326.

CALGB 9160: Budman, D., Rosner, G., Lichtman, S., Miller, A., Ratain, M. and Schilsky, R. (1998) A randomized trial of wr-2721 (amifostine) as a chemoprotective agent in combination with high-dose cyclophosphamide and molgramostim (GM-CSG). Cancer Therapeutics, 1: 164-167.

References

Mueller, P. and Rosner, G. (1998). Semiparametric PK/PD Models. In: Practical Nonparametric and Semiparametric Bayesian Statistics, Eds: D. Dey, P. Muller, D. Sinha, New York: Springer-Verlag, pp. 323-337.

Mueller, P., Quintana, F. and Rosner, G. (2004). A Method for Combining Inference over Related Nonparametric Bayesian Models. Journal of the Royal Statistical Society, Series B, 66: 735-749.

Examples

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

`calgb.pred` *Cancer and Leukemia Group B (CALGB) Data for Prediction*

Description

Same as 'calgb' for future patients (for prediction).

Usage

```
data(calgb.pred)
```

Format

A data frame with 8 observations on the following 10 variables.

- `Z1` a numeric vector giving the estimated Z1 coefficients of the logistic regression curve.
- `Z2` a numeric vector giving the estimated Z2 coefficients of the logistic regression curve.
- `Z3` a numeric vector giving the estimated Z3 coefficients of the logistic regression curve.
- `T1` a numeric vector giving the estimated time point where the horizontal line of the curve is defined, i.e., the curve consists of a horizontal line up to t=T1ji.
- `T2` a numeric vector giving the estimated time point where the logistic component of the curve is defined, i.e., the curve consist of a logistic regression curve starting at t=T2ji.
- `B0` a numeric vector giving the estimated B0 coefficients of the logistic regression curve.

B1 a numeric vector giving the estimated B1 coefficients of the logistic regression curve.

CTX a numeric vector giving the dose level of cyclophosphamide.

GM a numeric vector giving the dose level GM-CSF.

AMOF a numeric vector giving the dose level of amifostine.

Examples

```
data(calgb.pred)
## maybe str(calgb.pred) ; plot(calgb.pred) ...
```

CSDPbinary *Bayesian analysis for a semiparametric logistic regression model*

Description

This function generates a posterior density sample for a semiparametric binary regression model using a Centrally Standarized Dirichlet process prior for the link function.

Usage

```
CSDPbinary(formula,baseline="logistic",prior,mcmc,state,status,misc=NULL,
         data=sys.frame(sys.parent()),na.action=na.fail)
```

Arguments

formula	a two-sided linear formula object describing the model fit, with the response on the left of a ~ operator and the terms, separated by + operators, on the right.
baseline	a description of the baseline error distribution to be used in the model. The baseline distributions considered by CSDPbinary so far is *logistic*.
prior	a list giving the prior information. The list includes the following parameters: *a0* and *b0* giving the hyperparameters for prior distribution of the precision parameter of the Centrally-Standarized Dirichlet process prior, *alpha* giving the value of the precision parameter (it must be specified if *a0* and *b0* are missing, see the details below), and *beta0* and *Sbeta0* giving the hyperparameters of the normal prior distribution for the regression coefficients.
mcmc	a list giving the MCMC parameters. The list must include the following integers: *nburn* giving the number of burn-in scans, *nskip* giving the thinning interval, *nsave* giving the total number of scans to be saved, *ntheta* giving the thinning interval for the *theta* parameter (if missing, the value 1 is considered), *ndisplay* giving the number of saved scans to be displayed on the screen (the function reports on the screen when every *ndisplay* iterations have been carried out), and *tune* giving the Metropolis tuning parameter (the default value is 1.1).
state	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.

`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object *state*.
`misc`	misclassification information. When used, this list must include two objects, *sens* and *spec*, giving the sensitivity and specificity, respectively. Both can be a vector or a scalar. This information is used to correct for misclassification in the conditional bernoulli model.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `CSDPbinary` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a semiparametric binary regression model using a Centrally-Standarized Dirichlet Process Prior (CSDP) (Newton, Czado and Chappell, 1996):

$$y_i = I(V_i \leq X_i\beta),\ i = 1, \ldots, n$$

$$V_1, \ldots, V_n | G \sim G$$

$$G | m, p, d, h \sim CSDP(m, p, d, h)$$

where, $m = \{m_1, m_2, m_3, m_4\}$ is the base measure, $m_j\,(B) = \alpha G_0\,(B)\, I\{A_j\,(\theta)\},\ j = 1, \ldots, 4,$

$$A_1\,(\theta) = (-\infty, \theta - d], A_2\,(\theta) = (\theta - d, 0]$$

$$A_3\,(\theta) = (0, \theta], A_4\,(\theta) = (\theta, \infty),$$

and h is a uniform distribution on $(0, d)$. Note that in the construction of Newton et al. (1996), $G = \frac{1-p}{2}(G_1 + G_4) + \frac{p}{2}(G_2 + G_3)$, where G_j are conditionally independent Dirichlet processes with base measure m_j.

To complete the model specification, the following prior distributions are assumed,

$$\alpha | a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\beta | \beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

The precision parameter, α, of the *CSDP* prior can be considered as random, having a *Gamma* distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random a strategy similar to the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

A Metropolis-Hastings step is used to sample the fully conditional distribution of the regression coefficients and errors (see, Jara, Garcia-Zattera and Lesaffre, 2006). In the computational implementation of the model, G is considered as latent data and sampled partially with sufficient accuracy to be able to generate $V_1, \ldots, V_{n+1}$ which are exactly iid G, as proposed by Doss (1994). Both Ferguson's definition of DP and the Sethuraman-Tiwari (1982) representation of the process are used, as described in Jara, Garcia-Zattera and Lesaffre (2006). An extra step which moves the clusters in such a way that the posterior distribution is still a stationary distribution, is performed in order to improve the rate of mixing.

Value

An object of class `CSDPbinary` representing the semiparametric logistic regression model fit. Generic functions such as `print`, `plot`, `predict`, `summary`, and `anova` have methods to show the results of the fit. The results include *beta*, the precision parameter (*alpha*), the number of clusters (*ncluster*), and the *link* function.

The MCMC samples of the parameters and the errors in the model are stored in the object *thetasave* and *randsave*, respectively. Both objects are included in the list *save.state* and are matrices which can be analyzed directly by functions provided by the coda package.

The list *state* in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set *status=TRUE* and create the list state based on this starting values. In this case the list *state* must include the following objects:

`beta`	giving the value of the regression coefficients.
`theta`	giving the value of the third quartile parameter.
`v`	giving the value of the errors (it must be consistent with *yi = I(Vi < xi beta)*.
,	
`y`	giving the value of the true response binary variable (only if the model considers correction for misclassification).
`alpha`	giving the value of the precision parameter.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

References

Doss, H. (1994) Bayesian nonparametric estimation for incomplete data using mixtures of Dirichlet priors. The Annals of Statistics, 22: 1763 - 1786.

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Jara, A., Garcia-Zattera, M.J., Lesaffre, E. (2006) Semiparametric Bayesian Analysis of Misclassified Binary Data. XXIII International Biometric Conference, July 16-21, Montréal, Canada.

Newton, M.A., Czado, C., and Chappell, R. (1996) Bayesian inference for semiparametric binary regression. Journal of the American Statistical Association, 91, 142-153.

Sethuraman, J., and Tiwari, R. C. (1982) Convergence of Dirichlet Measures and the Interpretation of their Parameter, in Statistical Decision Theory and Related Topics III (vol. 2), eds. S. S. Gupta and J. O. Berger, New York: Academic Press, pp. 305 - 315.

Examples

```
## Not run:
    # Bioassay Data Example
    # Cox, D.R. and Snell, E.J. (1989). Analysis of Binary Data. 2nd ed.
    # Chapman and Hall. p. 7.
    # In this example there are 150 subjects at 5 different stimulus
    # levels, 30 at each level.


      y<-c(rep(0,30-2),rep(1,2),
           rep(0,30-8),rep(1,8),
           rep(0,30-15),rep(1,15),
           rep(0,30-23),rep(1,23),
           rep(0,30-27),rep(1,27))

      x<-c(rep(0,30),
           rep(1,30),
           rep(2,30),
           rep(3,30),
           rep(4,30))


    # Initial state
      state <- NULL

    # MCMC parameters
      nburn<-5000
      nsave<-10000
      nskip<-10
      ntheta<-1
      ndisplay<-100
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,
                   ntheta=ntheta,ndisplay=ndisplay,tune=1.1)


    # Prior distribution
      prior <- list(alpha=1, d=2*log(3), p=0.5, beta0=rep(0,2),
                    Sbeta0=diag(1000,2))

    # Fitting the model

      fit1 <- CSDPbinary(y~x,prior=prior,mcmc=mcmc,state=state,
                         status=TRUE)
      fit1

    # Summary with HPD and Credibility intervals
      summary(fit1)
      summary(fit1,hpd=FALSE)

    # Plot model parameters (to see the plots gradually set ask=TRUE)
      plot(fit1)

    # Plot an specific model parameter (to see the plots gradually
    # set ask=TRUE)
      plot(fit1,ask=FALSE,nfigr=1,nfigc=2,param="x")
      plot(fit1,ask=FALSE,param="link",nfigc=1,nfigr=1)

    # Table of Pseudo Contour Probabilities
```

```
    anova(fit1)

  # Predictive Distribution
    npred<-40
    xnew<-cbind(rep(1,npred),seq(0,4,length=npred))

    pp<-predict(fit1,xnew)

    plot(seq(0,4,length=npred),pp$pmean,type='l',ylim=c(0,1),
         xlab="log2(concentration)",ylab="Probability")

  # Adding MLE estimates
    points(c(0,1,2,3,4),c(0.067,0.267,0.500,0.767,0.900),col="red")


## End(Not run)
```

`deterioration` *Time to Cosmetic Deterioration of Breast Cancer Patients*

Description

This data set considers information of time to cosmetic deterioration of the breast for women with Stage 1 breast cancer who have undergone a lumpectomy, for two treatments, these being radiation, and radiation coupled with chemotherapy. There is interest in the cosmetic impact of the treatments because both are considered very effective in preventing recurrence of this early stage cancer. The data come from a retrospective study of 46 patients who received radiation only and 48 who received radiation plus chemotherapy. Each woman made a series of visits to a clinician, who determined whether or not retraction had occurred. If it had, the time of retraction was known only to lie between the time of the present and last visits. The data set is presented in Beadle et al. (1984a,b) and also given in Finkelstein and Wolfe (1985).

Usage

```
data(deterioration)
```

Format

A data frame with 94 observations on the following 3 variables.

`left`	a numeric vector giving the left limit of the interval
`right`	a numeric vector giving the right limit of the interval
`trt`	a numeric vector giving the treatment (0 = radiation only, 1 = radiation plus chemotherapy)

Source

Beadle, G., Come, S., Henderson, C., Silver, B., and Hellman, S. (1984a). The effect of adjuvant chemotherapy on the cosmetic results after primary radiation treatment for early stage breast cancer. International Journal of Radiation Oncology, Biology and Physics, 10: 2131-2137.

Beadle, G., Harris, J., Silver, B., Botnick, L., and Hellman, S. (1984b). Cosmetic results following primary radiation therapy for early breast cancer. Cancer, 54: 2911-2918.

Finkelstein, D.M. and Wolfe, R.A. (1985). A semiparametric model for regression analysis of interval-censored failure time data. Biometrics, 41: 933-945.

References

Hanson, T., and Johnson, W. (2004) A Bayesian Semiparametric AFT Model for Interval-Censored Data. Journal of Computational and Graphical Statistics, 13: 341-361.

Examples

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

`DPbetabinom` *Bayesian Semiparametric Beta-Binomial Model using a DP prior*

Description

This function generates a posterior density sample for a semiparametric version of the Beta-Binomial model using a Dirichlet process prior for the mixing distribution.

Usage

```
DPbetabinom(y,ngrid,prior,mcmc,state,status,
          data=sys.frame(sys.parent()),work.dir=NULL)
```

Arguments

`y`	a matrix giving the binomial data. The first column must include the number of sucess and the second column the number of trials.
`ngrid`	number of grid points where the predictive density estimate is evaluated.
`prior`	a list giving the prior information. The list includes the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing, see details below), and `a1` and `b1` giving the parameters of the beta centering distribution.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).

`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`work.dir`	working directory.

Details

This generic function fits a semiparametric version of the Beta-Binomial model (Liu, 1996):

$$y_i|n_i, p_i \sim Binom(n_i, p_i), i = 1, \ldots, n$$

$$p_i|G \sim G$$

$$G|\alpha, G_0 \sim DP(\alpha G_0)$$

where, the baseline distribution is the beta distribution,

$$G_0 = Beta(a_1, b_1)$$

To complete the model specification, the following hyperprior can be assumed for the total mass parameter:

$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$

Notice that the baseline distribution, G_0, is a conjugate prior in this model specification. Therefore, standard algorihtms for conjugate DP models are used (see, e.g., Escobar and West, 1995; MacEachern, 1998).

Value

An object of class `DPbetabinom` representing the DP Beta-Binomial model fit. Generic functions such as `print`, `summary`, and `plot` have methods to show the results of the fit. The results include the baseline parameters, `alpha`, and the number of clusters.

The MCMC samples of the parameters in the model are stored in the object `thetasave`. The object is included in the list `save.state` and are matrices which can be analyzed directly by functions provided by the coda package. The subject-specific binomial probabilities are stored in the object `randsave`.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`ncluster`	an integer giving the number of clusters.
`p`	a vector of dimension (no. observations+1) giving the current value of the binomial probabilities.
`ss`	an interger vector defining to which of the `ncluster` clusters each observation belongs.
`alpha`	giving the value of the precision parameter.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

Fernando Quintana <<`quintana@mat.puc.cl`>>

References

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Liu, J.S. (1996). Nonparametric Hierarchical Bayes via Sequential Imputations. The Annals of Statistics, 24: 911-930.

MacEachern, S.N. (1998) Computational Methods for Mixture of Dirichlet Process Models, in Practical Nonparametric and Semiparametric Bayesian Statistics, eds: D. Dey, P. Muller, D. Sinha, New York: Springer-Verlag, pp. 1-22.

Examples

```
## Not run:
    # Data
      data(rolling)
      y <- cbind(rolling$y1,rolling$y2)


    # Prior information

      prior<-list(alpha=1,
                  a1=1,
                  b1=1)

    # Initial state
      state <- NULL

    # MCMC parameters

      mcmc <- list(nburn=5000,
                   nsave=10000,
                   nskip=3,
                   ndisplay=100)

    # Fitting the model

      fit <- DPbetabinom(y=y,ngrid=100,
                         prior=prior,
                         mcmc=mcmc,
                         state=state,
                         status=TRUE)

      fit
      summary(fit)

    # density estimate
      plot(fit,output="density")

    # parameters
      plot(fit,output="param")

## End(Not run)
```

DPbinary *Bayesian analysis for a semiparametric Bernoulli regression model*

Description

This function generates a posterior density sample for a semiparametric binary regression model.

Usage

```
DPbinary(formula,inter=TRUE,baseline="logistic",prior,mcmc,state,
          status,misc=NULL,data=sys.frame(sys.parent()),
          na.action=na.fail)
```

Arguments

`formula`	a two-sided linear formula object describing the model fit, with the response on the left of a ~ operator and the terms, separated by + operators, on the right.
`inter`	a logical variable indicating whether the intercept should be explicitly included in the linear predictor `TRUE` or absorved in the unknown link function `FALSE`. The default option is `TRUE`. Note that if inter is `FALSE` the dimension of the prior distribution for the regression coefficients should not consider the intercept in the model.
`baseline`	a description of the baseline error distribution to be used in the model. The baseline distributions considered by `DPbinary` so far are *logistic* (default), *normal*, and *cauchy*.
`prior`	a list giving the prior information. The list includes the following parameters: *a0* and *b0* giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, *alpha* giving the value of the precision parameter (it must be specified if *a0* and *b0* are missing, see the details below), and *beta0* and *Sbeta0* giving the hyperparameters of the normal prior distribution for the regression coefficients.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: *nburn* giving the number of burn-in scans, *nskip* giving the thinning interval, *nsave* giving the total number of scans to be saved, *ndisplay* giving the number of saved scans to be displayed on the screen (the function reports on the screen when every *ndisplay* iterations have been carried out), and *tune* giving the Metropolis tuning parameter (the default value is 1.1).

`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object *state*.
`misc`	misclassification information. When used, this list must include two objects, *sens* and *spec*, giving the sensitivity and specificity, respectively. Both can be a vector or a scalar. This information is used to correct for misclassification in the conditional bernoulli model.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `DPbinary` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a semiparametric binary regression model using a Dirichlet process prior (see, Jara, Garcia-Zattera and Lesaffre, 2006):

$$y_i = I(V_i \leq X_i\beta), i = 1, \ldots, n$$

$$V_1, \ldots, V_n | G \sim G$$

$$G|\alpha, G_0 \sim DP(\alpha G_0)$$

where, $G_0 = Logistic(V|0,1)$ if the baseline is logistic, $G_0 = N(V|0,1)$ if the baseline is normal, and $G_0 = Cauchy(V|0,1)$ if the baseline is cauchy. To complete the model specification, the following prior distributions are assumed,

$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\beta|\beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

The precision or total mass parameter, α, of the *DP* prior can be considered as random, having a *gamma* distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

A Metropolis-Hastings step is used to sample the fully conditional distribution of the regression coefficients and errors (see, Jara, Garcia-Zattera and Lesaffre, 2006). In the computational implementation of the model, G is considered as latent data and sampled partially with sufficient accuracy to be able to generate $V_1, \ldots, V_{n+1}$ which are exactly iid G, as proposed by Doss (1994). Both Ferguson's definition of DP and the Sethuraman-Tiwari (1982) representation of the process are used, as described in Jara, Garcia-Zattera and Lesaffre (2006). An extra step which moves the clusters in such a way that the posterior distribution is still a stationary distribution, is performed in order to improve the rate of mixing.

Value

An object of class `DPbinary` representing the semiparametric logistic regression model fit. Generic functions such as `print`, `plot`, `predict`, `summary`, and `anova` have methods to show the results of the fit. The results include `beta`, the precision parameter (*alpha*), the number of clusters (*ncluster*), and the *link* function.

The MCMC samples of the parameters and the errors in the model are stored in the object *thetasave* and *randsave*, respectively. Both objects are included in the list *save.state* and are matrices which can be analyzed directly by functions provided by the coda package.

The list *state* in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set *status=TRUE* and create the list state based on this starting values. In this case the list *state* must include the following objects:

`beta`	giving the value of the regression coefficients.
`v`	giving the value of the errors (it must be consistent with *yi = I(Vi < xi beta)*.
,	
`y`	giving the value of the true response binary variable (only if the model considers correction for misclassification).
`alpha`	giving the value of the precision parameter.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

References

Doss, H. (1994) Bayesian nonparametric estimation for incomplete data using mixtures of Dirichlet priors. The Annals of Statistics, 22: 1763 - 1786.

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Jara, A., Garcia-Zattera, M.J., Lesaffre, E. (2006) Semiparametric Bayesian Analysis of Misclassified Binary Data. XXIII International Biometric Conference, July 16-21, Montréal, Canada.

Sethuraman, J., and Tiwari, R. C. (1982) Convergence of Dirichlet Measures and the Interpretation of their Parameter, in Statistical Decision Theory and Related Topics III (vol. 2), eds. S. S. Gupta and J. O. Berger, New York: Academic Press, pp. 305 - 315.

Examples

```
## Not run:
    # Bioassay Data Example
    # Cox, D.R. and Snell, E.J. (1989). Analysis of Binary Data. 2nd ed.
    # Chapman and Hall. p. 7
    # In this example there are 150 subjects at 5 different stimulus levels,
    # 30 at each level.

      y<-c(rep(0,30-2),rep(1,2),
           rep(0,30-8),rep(1,8),
           rep(0,30-15),rep(1,15),
           rep(0,30-23),rep(1,23),
           rep(0,30-27),rep(1,27))

      x<-c(rep(0,30),
           rep(1,30),
           rep(2,30),
           rep(3,30),
           rep(4,30))

    # Initial state
      state <- NULL

    # MCMC parameters
      nburn<-5000
      nsave<-10000
      nskip<-10
      ndisplay<-100
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay,
                   tune=1.1)


    # Prior distribution
      prior <- list(a0=2,b0=1,beta0=rep(0,2), Sbeta0=diag(10000,2))

    # Fit the model
      fit1 <- DPbinary(y~x,prior=prior,mcmc=mcmc,state=state,status=TRUE)
      fit1

    # Summary with HPD and Credibility intervals
      summary(fit1)
      summary(fit1,hpd=FALSE)

    # Plot model parameters (to see the plots gradually set ask=TRUE)
      plot(fit1)
      plot(fit1,nfigr=2,nfigc=2)

    # Plot an specific model parameter (to see the plots gradually
    # set ask=TRUE)
      plot(fit1,ask=FALSE,nfigr=1,nfigc=2,param="x")
      plot(fit1,ask=FALSE,nfigr=1,nfigc=2,param="ncluster")
      plot(fit1,ask=FALSE,param="link",nfigc=1,nfigr=1)

    # Table of Pseudo Contour Probabilities
      anova(fit1)

    # Predictive Distribution
      npred<-40
      xnew<-cbind(rep(1,npred),seq(0,4,length=npred))

      pp<-predict(fit1,xnew)
```

```
        plot(seq(0,4,length=npred),pp$pmean,type='l',ylim=c(0,1),
             xlab="log2(concentration)",ylab="Probability")

      # Adding MLE estimates
        points(c(0,1,2,3,4),c(0.067,0.267,0.500,0.767,0.900),col="red")


## End(Not run)
```

DPcaterpillar *Caterpillar Plots for Random Effects*

Description

This generic function produces Caterpillar Plots for Random Effects from DPrandom objects.

Usage

```
DPcaterpillar(object, midpoint="mean", hpd=TRUE , ask=TRUE,
              nfigr=1, nfigc=1, ...)
```

Arguments

`object`	DPrandom object from which random effects estimates can be extracted.
`midpoint`	variable indicating whether the mean or median of the posterior distribution of random effects should be considered as "midpoint" in the caterpillar plot.
`hpd`	logical variable indicating whether the hpd (TRUE) or pd (FALSE) of random effects should be considered in the caterpillar plot.
`ask`	logical variable indicating whether the caterpillar plots should be display gradually (TRUE) or not (FALSE).
`nfigr`	integer variable indicating the number of caterpillar plots by row.
`nfigc`	integer variable indicating the number of caterpillar plots by column.
`...`	further arguments passed to or from other methods.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

Examples

```
## Not run:
    # School Girls Data Example

      data(schoolgirls)
      attach(schoolgirls)

    # Prior information
    # Prior information

      tinv<-diag(10,2)
      prior<-list(alpha=1,nu0=4.01,tau1=0.001,tau2=0.001,
      tinv=tinv,mub=rep(0,2),Sb=diag(1000,2))

    # Initial state
      state <- NULL

    # MCMC parameters

      nburn<-5000
      nsave<-25000
      nskip<-20
      ndisplay<-1000
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,
                   ndisplay=ndisplay)

    # Fit the model

      fit1<-DPlmm(fixed=height~1,random=~age|child,prior=prior,
                 mcmc=mcmc,state=state,status=TRUE)
      fit1


    # Extract random effects

      DPcaterpillar(DPrandom(fit1))

## End(Not run)
```

DPcdensity *Bayesian Semiparametric Conditional Density Estimation using a DPM of normals*

Description

This function generates a posterior density sample for a Bayesian density regression model with continuous predictors using a Dirichlet process mixture of normals model.

Usage

```
DPcdensity(y,x,xpred,ngrid=100,grid=NULL,compute.band=FALSE,
           type.band="PD",prior,mcmc,state,status,
           data=sys.frame(sys.parent()),work.dir=NULL)
```

Arguments

y	a vector giving the data from which the density estimate is to be computed.

`x`	a vector or matrix giving the continuous predictors of dimension `nrec` times `nx`.
`xpred`	a vector or matrix giving the values of the continuous predictors used for prediction.
`ngrid`	number of grid points where the conditional density estimate is evaluated. The default is 100.
`grid`	vector of grid points where the conditional density estimate is evaluated. The default value is NULL and the grid is chosen according to the range of the data.
`compute.band`	logical variable indicating whether the credible band for the conditional density and mean function must be computed.
`type.band`	string indication the type of credible band to be computed; if equal to "HPD" or "PD" then the 95 percent pointwise HPD or PD band is computed, respectively.
`prior`	a list giving the prior information. The list includes the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing, see details below), `nu2` and `psiinv2` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix, `Psi1`, of the inverted Wishart part of the baseline distribution, `tau1` and `tau2` giving the hyperparameters for the gamma prior distribution of the scale parameter `k0` of the normal part of the baseline distribution, `m2` and `s2` giving the mean and the covariance of the normal prior for the mean, `m1`, of the normal component of the baseline distribution, respectively, `nu1` and `psiinv1` (it must be specified if `nu2` is missing, see details below) giving the hyperparameters of the inverted Wishart part of the baseline distribution and, `m1` giving the mean of the normal part of the baseline distribution (it must be specified if `m2` is missing, see details below) and, `k0` giving the scale parameter of the normal part of the baseline distribution (it must be specified if `tau1` is missing, see details below). Notice that the dimension of the baseline measure includes the predictor and the response, i.e., `nx+1`.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`work.dir`	working directory.

Details

This generic function fits a Dirichlet process mixture of normal model (Escobar and West, 1995) for the conditional density estimation $f(y \mid x)$ as proposed by Muller, Erkanli and West (1996). They proposed to specify a Dirichlet process mixture of normals for the joint distribution of the response and predictors. Although in the original paper these authors focussed on the mean regression function, their method can be used to model the conditional density of the response giving the predictors in a semiparametric way. Indeed, their method is essentially a locally weighted mixture of normal regression models with weigths predictor-dependent.

Let y_i and X_i be the response and the vector of predictors, respectively. Further, let $Z_i = (y_i, X_i)$. The model for the joint distribution of the response and predictors is as follows:

$$Z_i|\mu_i, \Sigma_i \sim N(\mu_i, \Sigma_i), i = 1, \ldots, n$$

$$(\mu_i, \Sigma_i)|G \sim G$$

$$G|\alpha, G_0 \sim DP(\alpha G_0)$$

where, the baseline distribution is the conjugate normal-inverted-Wishart,

$$G_0 = N(\mu|m_1, (1/k_0)\Sigma)IW(\Sigma|\nu_1, \psi_1)$$

To complete the model specification, independent hyperpriors are assumed (optional),

$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$

$$m_1|m_2, s_2 \sim N(m_2, s_2)$$

$$k_0|\tau_1, \tau_2 \sim Gamma(\tau_1/2, \tau_2/2)$$

$$\psi_1|\nu_2, \psi_2 \sim IW(\nu_2, \psi_2)$$

Note that the inverted-Wishart prior is parametrized such that if $A \sim IW_q(\nu, \psi)$ then $E(A) = \psi^{-1}/(\nu - q - 1)$.

To let part of the baseline distribution fixed at a particular value, set the corresponding hyperparameters of the prior distributions to NULL in the hyperprior specification of the model.

Although the baseline distribution, G_0, is a conjugate prior in this model specification, the algorithms with auxiliary parameters described in Neal (2000) is adopted. Specifically, the algorithm 8 with $m = 1$ of Neal (2000) is considered in the `DPcdensity` function.

Value

An object of class `DPcdensity` representing the DP mixture of normals model fit. Generic functions such as `print`, `summary`, and `plot` have methods to show the results of the fit. The results include the baseline parameters, `alpha`, and the number of clusters.

The MCMC samples of the parameters and the errors in the model are stored in the object `thetasave`. The object is included in the list `save.state` and are matrices which can be analyzed directly by functions provided by the coda package.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`ncluster`	an integer giving the number of clusters.

`muclus`	a matrix of dimension (nobservations+2)*(nvariables) giving the means of the clusters (only the first `ncluster` are considered to start the chain).
`sigmaclus`	a matrix of dimension (nobservations+2)*((nvariables)*((nvariables)+1)/2) giving the lower matrix of the covariance matrix of the clusters (only the first `ncluster` are considered to start the chain).
`ss`	an interger vector defining to which of the `ncluster` clusters each observation belongs.
`alpha`	giving the value of the precision parameter.
`m1`	giving the mean of the normal components of the baseline distribution.
`k0`	giving the scale parameter of the normal part of the baseline distribution.
`psi1`	giving the scale matrix of the inverted-Wishart part of the baseline distribution.
`z`	giving the matrix of response and predictors. This must be included if missing data (response and/or predictors) are present. Those are imputed during the MCMC.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

Tim Hanson <<`hansont@stat.sc.edu`>>

References

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Muller, P., Erkanli, A. and West, M. (1996) Bayesian curve fitting using multivariate normal mixtures. Biometrika, 83: 67-79.

Neal, R. M. (2000). Markov Chain sampling methods for Dirichlet process mixture models. Journal of Computational and Graphical Statistics, 9: 249-265.

See Also

`DPdensity`, `PTdensity`, `BDPdensity`

Examples

```
## Not run:

    ##########################################################
    # Simulated data:
    # Here we replicate the results reported with
    # simulated data by Dunson, Pillai and Park (2007,
    # JRSS Ser. B, 69: 163-183, pag 177) where a different
    # approach is proposed.
    ##########################################################

      dtrue <- function(grid,x)
      {
          exp(-2*x)*dnorm(grid,mean=x,sd=sqrt(0.01))+
          (1-exp(-2*x))*dnorm(grid,mean=x^4,sd=sqrt(0.04))
      }

      nobs <- 500
      x <- runif(nobs)
      y1 <- x + rnorm(nobs, 0, sqrt(0.01))
      y2 <- x^4 + rnorm(nobs, 0, sqrt(0.04))

      u <- runif(nobs)
      prob <- exp(-2*x)
      y <- ifelse(u<prob,y1,y2)

    # Prior information
      w <- cbind(y,x)
      wbar <- apply(w,2,mean)
      wcov <- var(w)

      prior <- list(a0=10,
                    b0=1,
                    nu1=4,
                    nu2=4,
                    s2=0.5*wcov,
                    m2=wbar,
                    psiinv2=2*solve(wcov),
                    tau1=6.01,
                    tau2=2.01)

    # Initial state
      state <- NULL

    # MCMC parameters

      mcmc <- list(nburn=5000,
                   nsave=5000,
                   nskip=3,
                   ndisplay=100)

    # fitting the model
      xpred <- c(0.00,0.05,0.10,0.15,0.20,0.25,
                 0.30,0.35,0.40,0.45,0.49,0.55,
                 0.60,0.65,0.70,0.75,0.80,0.85,
                 0.88,0.95,1.00)

      fit <- DPcdensity(y=y,x=x,xpred=xpred,ngrid=100,
                        prior=prior,
                        mcmc=mcmc,
                        state=state,
                        status=TRUE,
                        compute.band=TRUE,type.band="PD")

    # true model and estimates
      par(mfrow=c(2,3))
      plot(fit$grid,fit$densp.h[3,],lwd=1,type="l",lty=2,
           main="x=0.10",xlab="values",ylab="density",ylim=c(0,4))
      lines(fit$grid,fit$densp.l[3,],lwd=1,type="l",lty=2)
      lines(fit$grid,fit$densp.m[3,],lwd=2,type="l",lty=1)
      lines(fit$grid,dtrue(fit$grid,xpred[3]),lwd=2,
            type="l",lty=1,col="red")

      plot(fit$grid,fit$densp.h[6,],lwd=1,type="l",lty=2,
           main="x=0.25",xlab="values",ylab="density",ylim=c(0,4))
      lines(fit$grid,fit$densp.l[6,],lwd=1,type="l",lty=2)
      lines(fit$grid,fit$densp.m[6,],lwd=2,type="l",lty=1)
      lines(fit$grid,dtrue(fit$grid,xpred[6]),lwd=2,
            type="l",lty=1,col="red")

      plot(fit$grid,fit$densp.h[11,],lwd=1,type="l",lty=2,
           main="x=0.49",xlab="values",ylab="density",ylim=c(0,4))
      lines(fit$grid,fit$densp.l[11,],lwd=1,type="l",lty=2)
      lines(fit$grid,fit$densp.m[11,],lwd=2,type="l",lty=1)
      lines(fit$grid,dtrue(fit$grid,xpred[11]),lwd=2,type="l",
            lty=1,col="red")

      plot(fit$grid,fit$densp.h[16,],lwd=1,type="l",lty=2,
           main="x=0.75",xlab="values",ylab="density",ylim=c(0,4))
      lines(fit$grid,fit$densp.l[16,],lwd=1,type="l",lty=2)
      lines(fit$grid,fit$densp.m[16,],lwd=2,type="l",lty=1)
      lines(fit$grid,dtrue(fit$grid,xpred[16]),lwd=2,type="l",
            lty=1,col="red")

      plot(fit$grid,fit$densp.h[19,],lwd=1,type="l",lty=2,
           main="x=0.75",xlab="values",ylab="density",ylim=c(0,4))
      lines(fit$grid,fit$densp.l[19,],lwd=1,type="l",lty=2)
      lines(fit$grid,fit$densp.m[19,],lwd=2,type="l",lty=1)
      lines(fit$grid,dtrue(fit$grid,xpred[19]),lwd=2,type="l",
            lty=1,col="red")

    # data and mean function
      plot(x,y,xlab="x",ylab="y",main="")
      lines(xpred,fit$meanfp.m,type="l",lwd=2,lty=1)
      lines(xpred,fit$meanfp.l,type="l",lwd=2,lty=2)
      lines(xpred,fit$meanfp.h,type="l",lwd=2,lty=2)


## End(Not run)
```

DPdensity — *Semiparametric Bayesian density estimation using a DPM of normals*

Description

This function generates a posterior density sample for a Dirichlet process mixture of normals model.

Usage

```
DPdensity(y,ngrid=1000,grid=NULL,prior,mcmc,state,status,
          method="neal",data=sys.frame(sys.parent()),
          na.action=na.fail)
```

Arguments

`y`	a vector or matrix giving the data from which the density estimate is to be computed.
`ngrid`	number of grid points where the density estimate is evaluated. This is only used if dimension of `y` is lower or equal than 2. The default value is 1000.
`grid`	matrix of dimension ngrid*nvar of grid points where the density estimate is evaluated. This is only used if dimension of `y` is lower or equal than 2. The default value is NULL and the grid is chosen according to the range of the data.
`prior`	a list giving the prior information. The list includes the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing, see details below), `nu2` and `psiinv2` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix, `Psi1`, of the inverted Wishart part of the baseline distribution, `tau1` and `tau2` giving the hyperparameters for the gamma prior distribution of the scale parameter `k0` of the normal part of the baseline distribution, `m2` and `s2` giving the mean and the covariance of the normal prior for the mean, `m1`, of the normal component of the baseline distribution, respectively, `nu1` and `psiinv1` (it must be specified if `nu2` is missing, see details below) giving the hyperparameters of the inverted Wishart part of the baseline distribution and, `m1` giving the mean of the normal part of the baseline distribution (it must be specified if `m2` is missing, see details below) and, `k0` giving the scale parameter of the normal part of the baseline distribution (it must be specified if `tau1` is missing, see details below).
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`method`	the method to be used. See `Details`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `DPdensity` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a Dirichlet process mixture of normal model for density estimation (Escobar and West, 1995):

$$y_i|\mu_i,\Sigma_i \sim N(\mu_i,\Sigma_i), i=1,\ldots,n$$

$$(\mu_i,\Sigma_i)|G \sim G$$

$$G|\alpha,G_0 \sim DP(\alpha G_0)$$

where, the baseline distribution is the conjugate normal-inverted-Wishart,

$$G_0 = N(\mu|m_1,(1/k_0)\Sigma)IW(\Sigma|\nu_1,\psi_1)$$

To complete the model specification, independent hyperpriors are assumed (optional),

$$\alpha|a_0,b_0 \sim Gamma(a_0,b_0)$$

$$m_1|m_2,s_2 \sim N(m_2,s_2)$$

$$k_0|\tau_1,\tau_2 \sim Gamma(\tau_1/2,\tau_2/2)$$

$$\psi_1|\nu_2,\psi_2 \sim IW(\nu_2,\psi_2)$$

Note that the inverted-Wishart prior is parametrized such that if $A \sim IW_q(\nu,\psi)$ then $E(A) = \psi^{-1}/(\nu-q-1)$.

To let part of the baseline distribution fixed at a particular value, set the corresponding hyperparameters of the prior distributions to NULL in the hyperprior specification of the model.

Although the baseline distribution, G_0, is a conjugate prior in this model specification, the algorithms with auxiliary parameters described in MacEachern and Muller (1998) and Neal (2000) are adopted. Specifically, the no-gaps algorithm of MacEachern and Muller (1998), `"no-gaps"`, and the algorithm 8 with $m=1$ of Neal (2000), `"neal"`, are considered in the `DPdensity` function. The default method is the algorithm 8 of Neal.

Value

An object of class `DPdensity` representing the DP mixture of normals model fit. Generic functions such as `print`, `summary`, and `plot` have methods to show the results of the fit. The results include the baseline parameters, `alpha`, and the number of clusters.

The function `DPrandom` can be used to extract the posterior mean of the subject-specific means and covariance matrices.

The MCMC samples of the parameters and the errors in the model are stored in the object `thetasave` and `randsave`, respectively. Both objects are included in the list `save.state` and are matrices which can be analyzed directly by functions provided by the coda package.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`ncluster`	an integer giving the number of clusters.

`muclus`	a matrix of dimension (nobservations+100)*(nvariables) giving the means of the clusters (only the first `ncluster` are considered to start the chain).
`sigmaclus`	a matrix of dimension (nobservations+100)*((nvariables)*((nvariables)+1)/2) giving the lower matrix of the covariance matrix of the clusters (only the first `ncluster` are considered to start the chain).
`ss`	an interger vector defining to which of the `ncluster` clusters each observation belongs.
`alpha`	giving the value of the precision parameter.
`m1`	giving the mean of the normal components of the baseline distribution.
`k0`	giving the scale parameter of the normal part of the baseline distribution.
`psi1`	giving the scale matrix of the inverted-Wishart part of the baseline distribution.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

References

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

MacEachern, S. N. and Muller, P. (1998) Estimating mixture of Dirichlet Process Models. Journal of Computational and Graphical Statistics, 7 (2): 223-338.

Neal, R. M. (2000). Markov Chain sampling methods for Dirichlet process mixture models. Journal of Computational and Graphical Statistics, 9: 249-265.

See Also

`DPrandom`, `PTdensity`, `BDPdensity`

Examples

```
## Not run:
    ####################################
    # Univariate example
    ####################################

    # Data
      data(galaxy)
      galaxy <- data.frame(galaxy,speeds=galaxy$speed/1000)
      attach(galaxy)

    # Initial state
      state <- NULL

    # MCMC parameters

      nburn <- 1000
      nsave <- 10000
```

```
  nskip <- 10
  ndisplay <- 100
  mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay)

# Example of Prior information 1
# Fixing alpha, m1, and Psi1

  prior1 <- list(alpha=1,m1=rep(0,1),psiinv1=diag(0.5,1),nu1=4,
               tau1=1,tau2=100)


# Example of Prior information 2
# Fixing alpha and m1

  prior2 <- list(alpha=1,m1=rep(0,1),psiinv2=solve(diag(0.5,1)),
               nu1=4,nu2=4,tau1=1,tau2=100)


# Example of Prior information 3
# Fixing only alpha

  prior3 <- list(alpha=1,m2=rep(0,1),s2=diag(100000,1),
             psiinv2=solve(diag(0.5,1)),
             nu1=4,nu2=4,tau1=1,tau2=100)


# Example of Prior information 4
# Everything is random

  prior4 <- list(a0=2,b0=1,m2=rep(0,1),s2=diag(100000,1),
             psiinv2=solve(diag(0.5,1)),
             nu1=4,nu2=4,tau1=1,tau2=100)

# Fit the models

  fit1.1 <- DPdensity(y=speeds,prior=prior1,mcmc=mcmc,
                    state=state,status=TRUE)
  fit1.2 <- DPdensity(y=speeds,prior=prior2,mcmc=mcmc,
                    state=state,status=TRUE)
  fit1.3 <- DPdensity(y=speeds,prior=prior3,mcmc=mcmc,
                    state=state,status=TRUE)
  fit1.4 <- DPdensity(y=speeds,prior=prior4,mcmc=mcmc,
                    state=state,status=TRUE)

# Posterior means
  fit1.1
  fit1.2
  fit1.3
  fit1.4

# Plot the estimated density
  plot(fit1.1,ask=FALSE)
  plot(fit1.2,ask=FALSE)
  plot(fit1.3,ask=FALSE)
  plot(fit1.4,ask=FALSE)

# Extracting the density estimate
  cbind(fit1.1$x1,fit1.1$dens)
  cbind(fit1.2$x1,fit1.2$dens)
  cbind(fit1.3$x1,fit1.3$dens)
  cbind(fit1.4$x1,fit1.4$dens)

# Plot the parameters (only prior 2 for illustration)
# (to see the plots gradually set ask=TRUE)
  plot(fit1.2,ask=FALSE,output="param")

# Plot the a specific parameters
# (to see the plots gradually set ask=TRUE)
  plot(fit1.2,ask=FALSE,output="param",param="psi1-speeds",
       nfigr=1,nfigc=2)

# Extracting the posterior mean of the specific
# means and covariance matrices
# (only prior 2 for illustration)
  DPrandom(fit1.2)

# Ploting predictive information about the specific
# means and covariance matrices
# with HPD and Credibility intervals
# (only prior 2 for illustration)
# (to see the plots gradually set ask=TRUE)
  plot(DPrandom(fit1.2,predictive=TRUE),ask=FALSE)
  plot(DPrandom(fit1.2,predictive=TRUE),ask=FALSE,hpd=FALSE)

# Ploting information about all the specific means
# and covariance matrices
# with HPD and Credibility intervals
# (only prior 2 for illustration)
# (to see the plots gradually set ask=TRUE)
  plot(DPrandom(fit1.2),ask=FALSE,hpd=FALSE)


####################################
# Bivariate example
####################################

# Data
  data(airquality)
  attach(airquality)

  ozone <- Ozone**(1/3)
  radiation <- Solar.R

# Prior information

  s2 <- matrix(c(10000,0,0,1),ncol=2)
```

```
    m2 <- c(180,3)
    psiinv2 <- solve(matrix(c(10000,0,0,1),ncol=2))

    prior <- list(a0=1,b0=1/5,nu1=4,nu2=4,s2=s2,
                  m2=m2,psiinv2=psiinv2,tau1=0.01,tau2=0.01)

  # Initial state
    state <- NULL

  # MCMC parameters

    nburn <- 5000
    nsave <- 10000
    nskip <- 10
    ndisplay <- 1000
    mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay)

  # Fit the model
    fit1 <- DPdensity(y=cbind(radiation,ozone),prior=prior,mcmc=mcmc,
                      state=state,status=TRUE,na.action=na.omit)

  # Plot the estimated density
    plot(fit1)

  # Extracting the density estimate
    fit1$x1
    fit1$x2
    fit1$dens

## End(Not run)
```

DPelicit *Performs a prior elicitation for the precision parameter of a DP prior*

Description

This function performs a prior elicitation for the precision parameter of a DP prior. The function calculates:

1) the expected value and the standard deviation of the number of clusters, given the values of the parameters of the gamma prior for the precision parameter, `a0` and `b0`, or

2) the value of the parameters `a0` and `b0` of the `gamma` prior distribution for the precision parameter, `alpha`, given the prior expected number and the standard deviation of the number of clusters.

Usage

```
DPelicit(n,method='JGL',a0=NULL,b0=NULL,mean=NULL,std=NULL)
```

Arguments

`n`	number of observations which distribution follows a DP prior.
`method`	the method to be used. See `details`.
`a0`	hyperparameter for the `Gamma` prior distribution of the precision parameter of the Dirichlet process prior, `alpha ~ Gamma(a0,b0)`.
`b0`	hyperparameter for the `Gamma` prior distribution of the precision parameter of the Dirichlet process prior, $alpha\ Gamma(a0, b0)$.
`mean`	prior expected number of clusters when $alpha\ Gamma(a0, b0)$.
`std`	prior standard deviation for the number of clusters when $alpha\ Gamma(a0, b0)$.

Details

The methods supported by these functions are based on the fact that a priori `E(alpha) = a0/b0` and `Var(alpha) = a0/b0^2`, and an additional approximation based on Taylor series expansion.

The default method, `"JGL"`, is based on the exact value of the mean and the variance of the number of clusters given the precision parameter alpha (see, Jara, Garcia-Zatera and Lesaffre, 2007).

The Method `"KMQ"` is base on the Liu (1996) approximation to the expected value and the variance of the number of clusters given the precision parameter alpha (see, Kottas, Muller and Quintana, 2005).

Given the prior judgement for the mean and variance of the number of clusters, the equations are numerically solve for `a0` and `b0`. With this objective, the Newton-Raphson algorithm and the forward-difference approximation to Jacobian are used.

Author(s)

Alejandro Jara <`<atjara@uc.cl>`>

References

Jara, A., Garcia-Zattera, M.J., Lesaffre, E. (2007) A Dirichlet Process mixture model for the analysis of correlated binary responses. Computational Statistics and Data Analysis 51: 5402-5415.

Kottas, A., Muller, P., Quintana, F. (2005) Nonparametric Bayesian modeling for multivariate ordinal data, Journal of Computational and Graphical Statistics 14: 610-625.

Liu, J.S. (1996) Nonparametric Hierarchical Bayes via Sequential Imputations, The Annals of Statistics, 24: 911-930.

Examples

```
# Calculate the expected value and the standard deviation
# for the number of cluster given alpha ~ Gamma(a0,b0).

  DPelicit(200,a0=2.01,b0=2.01,method="JGL")
  DPelicit(200,a0=2.01,b0=2.01,method="KMQ")

# Calculate the values of a0 and b0, given the expected value
```

```
    # and the standard deviation of the number of clusters

      DPelicit(200,mean=3.1,std=2.7,method="JGL")
      DPelicit(200,mean=3.1,std=2.7,method="KMQ")
```

DPglmm *Bayesian analysis for a semiparametric generalized linear mixed model using a MDP*

Description

This function generates a posterior density sample for a semiparametric generalized linear mixed model using a Dirichlet Process or a Mixture of Dirichlet process prior for the distribution of the random effects.

Usage

```
DPglmm(fixed,random,family,offset,n,prior,mcmc,state,status,
      data=sys.frame(sys.parent()),na.action=na.fail)
```

Arguments

`fixed`	a two-sided linear formula object describing the fixed-effects part of the model, with the response on the left of a ~ operator and the terms, separated by + operators, on the right.
`random`	a one-sided formula of the form `~z1+...+zn | g`, with `z1+...+zn` specifying the model for the random effects and `g` the grouping variable. The random effects formula will be repeated for all levels of grouping.
`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. The families(links) considered by `DPglmm` so far are binomial(logit), binomial(probit), Gamma(log), and poisson(log). The gaussian(identity) case is implemented separately in the function `DPlmm`.
`offset`	this can be used to specify an a priori known component to be included in the linear predictor during the fitting (only for poisson and gamma models).
`n`	this can be used to indicate the total number of cases in a binomial model (only implemented for the logistic link). If it is not specified the response variable must be binary.
`prior`	a list giving the prior information. The list includes the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing, see details below), `nu0` and `Tinv` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix of the normal baseline distribution, `sigma` giving the value of the covariance matrix of the centering distribution (it must be specified if `nu0`

	and `tinv` are missing), `mub` and `Sb` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `mu` giving the value of the mean of the centering distribution (it must be specified if `mub` and `Sb` are missing), `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the fixed effects (must be specified only if fixed effects are considered in the model), and `tau1` and `tau2` giving the hyperparameters for the gamma prior distribution for the inverse of the precision parameter of the Gamma model (they must be specified only if the Gamma model is considered).
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, `ndisplay` giving the number of saved scans to be displayed on the screen (the function reports on the screen when every `ndisplay` iterations have been carried out), `tune1` giving the positive Metropolis tuning parameter for the precision parameter of the Gamma model (the default value is 1.1).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `DPglmm` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a generalized linear mixed-effects model, where the linear predictor is modeled as follows:

$$\eta_i = X_i \beta_F + Z_i \beta_R + Z_i b_i, i = 1, \ldots, n$$

$$\theta_i | G \sim G$$

$$G | \alpha, G_0 \sim DP(\alpha G_0)$$

where, $\theta_i = \beta_R + b_i$, $\beta = \beta_F$, and $G_0 = N(\theta | \mu, \Sigma)$. To complete the model specification, independent hyperpriors are assumed,

$$\alpha | a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\beta | \beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

$$\mu | \mu_b, S_b \sim N(\mu_b, S_b)$$

$$\Sigma | \nu_0, T \sim IW(\nu_0, T)$$

Note that the inverted-Wishart prior is parametrized such that $E(\Sigma) = T^{-1}/(\nu_0 - q - 1)$.

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value set, a_0 to NULL in the prior specification.

The inverse of the dispersion parameter of the Gamma model is modeled using `gamma` distribution, $\Gamma(\tau_1/2, \tau_2/2)$.

The computational implementation of the model is based on the marginalization of the `DP` and the MCMC is model-specific.

For the `binomial(logit)`, `poisson`, and `Gamma`, MCMC methods for nonconjugate priors (see, MacEachern and Muller, 1998; Neal, 2000) are used. Specifically, the algorithm 8 with `m=1` of Neal (2000), is considered in the `DPglmm` function. In this case, the fully conditional distributions for fixed and in the resampling step of random effects are generated through the Metropolis-Hastings algorithm with a IWLS proposal (see, West, 1985 and Gamerman, 1997).

For conditonal bernoulli model `binomial(probit)` the following latent variable representation is used:

$$y_{ij} = I(w_{ij} > 0), j = 1, \ldots, n_i$$
$$w_{ij} | \beta, \theta_i, \lambda_i \sim N(X_{ij}\beta + Z_{ij}\theta_i, 1)$$

In this case, the computational implementation of the model is based on the marginalization of the `DP` and on the use of MCMC methods for conjugate priors (Escobar, 1994; Escobar and West, 1998).

The β_R parameters are sampled using the ϵ-DP approximation proposed by Muliere and Tardella (1998), with ϵ=0.01.

Value

An object of class `DPglmm` representing the generalized linear mixed-effects model fit. Generic functions such as `print`, `plot`, `summary`, and `anova` have methods to show the results of the fit. The results include `betaR`, `betaF`, `mu`, the elements of `Sigma`, the precision parameter `alpha`, the number of clusters, and the dispersion parameter of the Gamma model.

The function `DPrandom` can be used to extract the posterior mean of the random effects.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`ncluster`	an integer giving the number of clusters.
`alpha`	giving the value of the precision parameter.
`b`	a matrix of dimension (nsubjects)*(nrandom effects) giving the value of the random effects for each subject.
`bclus`	a matrix of dimension (nsubjects)*(nrandom effects) giving the value of the random effects for each clusters (only the first `ncluster` are considered to start the chain).
`ss`	an interger vector defining to which of the `ncluster` clusters each subject belongs.
`beta`	giving the value of the fixed effects.
`mu`	giving the mean of the normal baseline distributions.
`sigma`	giving the variance matrix of the normal baseline distributions.
`phi`	giving the precision parameter for the Gamma model (if needed).

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

References

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Escobar, M.D. and West, M. (1998) Computing Bayesian Nonparametric Hierarchical Models, in Practical Nonparametric and Semiparametric Bayesian Statistics, eds: D. Dey, P. Muller, D. Sinha, New York: Springer-Verlag, pp. 1-22.

Gamerman, D. (1997) Sampling from the posterior distribution in generalized linear mixed models. Statistics and Computing, 7: 57-68.

MacEachern, S. N. and Muller, P. (1998) Estimating mixture of Dirichlet Process Models. Journal of Computational and Graphical Statistics, 7 (2): 223-338.

Muliere, P. and Tardella, L. (1998) Approximating distributions of random functionals of Ferguson-Dirichlet priors. The Canadian Journal of Statistics, 26(2): 283-297.

Neal, R. M. (2000) Markov Chain sampling methods for Dirichlet process mixture models. Journal of Computational and Graphical Statistics, 9:249-265.

West, M. (1985) Generalized linear models: outlier accomodation, scale parameter and prior distributions. In Bayesian Statistics 2 (eds Bernardo et al.), 531-558, Amsterdam: North Holland.

See Also

`DPrandom`, `DPlmm` , `DPolmm` , `DPMlmm` ,`DPMglmm` , `DPMolmm`, `PTlmm` , `PTglmm`, `PTolmm`

Examples

```
## Not run:
    # Respiratory Data Example

      data(indon)
      attach(indon)

      baseage2<-baseage**2
      follow<-age-baseage
      follow2<-follow**2

    # Prior information

      beta0<-rep(0,9)
      Sbeta0<-diag(1000,9)
      tinv<-diag(1,1)
      prior<-list(a0=2,b0=0.1,nu0=4,tinv=tinv,mub=rep(0,1),Sb=diag(1000,1),
                  beta0=beta0,Sbeta0=Sbeta0)

    # Initial state
      state <- NULL
    # MCMC parameters

      nburn <- 5000
      nsave <- 5000
      nskip <- 0
      ndisplay <- 1000
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay)

    # Fit the Probit model
      fit1 <- DPglmm(fixed=infect~gender+height+cosv+sinv+xero+baseage+
                     baseage2+follow+follow2,random=~1|id,
                     family=binomial(probit),prior=prior,mcmc=mcmc,
                     state=state,status=TRUE)

    # Fit the Logit model
      fit2 <- DPglmm(fixed=infect~gender+height+cosv+sinv+xero+baseage+
                     baseage2+follow+follow2,random=~1|id,
                     family=binomial(logit),prior=prior,mcmc=mcmc,
                     state=state,status=TRUE)

    # Summary with HPD and Credibility intervals
      summary(fit1)
      summary(fit1,hpd=FALSE)

      summary(fit2)
      summary(fit2,hpd=FALSE)


    # Plot model parameters
    # (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE)
      plot(fit1,ask=FALSE,nfigr=2,nfigc=2)

    # Plot an specific model parameter
    # (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE,nfigr=1,nfigc=2,param="baseage")
      plot(fit1,ask=FALSE,nfigr=1,nfigc=2,param="ncluster")

## End(Not run)
```

DPlmm *Bayesian analysis for a semiparametric linear mixed model using a MDP*

Description

This function generates a posterior density sample for a semiparametric linear mixed model using a Dirichlet process or a Mixture of Dirichlet process prior for the distribution of the random effects.

Usage

```
DPlmm(fixed,random,prior,mcmc,state,status,
      data=sys.frame(sys.parent()),na.action=na.fail)
```

Arguments

`fixed`	a two-sided linear formula object describing the fixed-effects part of the model, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right.
`random`	a one-sided formula of the form `~z1+...+zn | g`, with `z1+...+zn` specifying the model for the random effects and `g` the grouping variable. The random effects formula will be repeated for all levels of grouping.
`prior`	a list giving the prior information. The list include the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` and `b0` are missing, see details below), `nu0` and `Tinv` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix of the normal baseline distribution, `sigma` giving the value of the covariance matrix of the centering distribution (it must be specified if `nu0` and `tinv` are missing), `mub` and `Sb` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `mu` giving the value of the mean of the centering distribution (it must be specified if `mub` and `Sb` are missing), `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the fixed effects (must be specified only if fixed effects are considered in the model) and, `tau1` and `tau2` giving the hyperparameters for the prior distribution of the error variance.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `DPlmm` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a linear mixed-effects model (Verbeke and Molenberghs, 2000):

$$y_i \sim N(X_i \beta_F + Z_i \beta_R + Z_i b_i, \sigma_e^2 I_{n_i}), i = 1, \ldots, n$$

$$\theta_i|G \sim G$$
$$G|\alpha, G_0 \sim DP(\alpha G_0)$$
$$\sigma_e^{-2}|\tau_1, \tau_2 \sim Gamma(\tau_1/2, \tau_2/2)$$

where, $\theta_i = \beta_R + b_i$, $\beta = \beta_F$, and $G_0 = N(\theta|\mu, \Sigma)$. To complete the model specification, independent hyperpriors are assumed,

$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$
$$\beta|\beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$
$$\mu|\mu_b, S_b \sim N(\mu_b, S_b)$$
$$\Sigma|\nu_0, T \sim IW(\nu_0, T)$$

Note that the inverted-Wishart prior is parametrized such that $E(\Sigma) = T^{-1}/(\nu_0 - q - 1)$.

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

The computational implementation of the model is based on the marginalization of the `DP` and on the use of MCMC methods for conjugate priors (Escobar, 1994; Escobar and West, 1998). The β_R parameters are sampled using the ϵ-DP approximation proposed by Muliere and Tardella (1998), with ϵ=0.01.

Value

An object of class `DPlmm` representing the linear mixed-effects model fit. Generic functions such as `print`, `plot`, `summary`, and `anova` have methods to show the results of the fit. The results include `betaR`, `betaF`, `sigma2e`, `mu`, the elements of `Sigma`, `alpha`, and the number of clusters.

The function `DPrandom` can be used to extract the posterior mean of the random effects.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`ncluster`	an integer giving the number of clusters.
`alpha`	giving the value of the precision parameter
`b`	a matrix of dimension (nsubjects)*(nrandom effects) giving the value of the random effects for each subject.
`bclus`	a matrix of dimension (nsubjects)*(nrandom effects) giving the value of the random effects for each clusters (only the first `ncluster` are considered to start the chain).
`ss`	an interger vector defining to which of the `ncluster` clusters each subject belongs.
`beta`	giving the value of the fixed effects.
`mu`	giving the mean of the normal baseline distributions.
`sigma`	giving the variance matrix of the normal baseline distributions.
`sigma2e`	giving the error variance.

Author(s)

Alejandro Jara <<atjara@uc.cl>>

References

Escobar, M.D. (1994) Estimating Normal Means with a Dirichlet Process Prior, Journal of the American Statistical Association, 89: 268-277.

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Escobar, M.D. and West, M. (1998) Computing Bayesian Nonparametric Hierarchical Models, in Practical Nonparametric and Semiparametric Bayesian Statistics, eds: D. Dey, P. Muller, D. Sinha, New York: Springer-Verlag, pp. 1-22.

Muliere, P. and Tardella, L. (1998) Approximating distributions of random functionals of Ferguson-Dirichlet priors. The Canadian Journal of Statistics, 26(2): 283-297.

Verbeke, G. and Molenberghs, G. (2000). Linear mixed models for longitudinal data, New York: Springer-Verlag.

See Also

DPrandom, DPglmm , DPolmm , DPMlmm , DPMglmm, DPMolmm, PTlmm , PTglmm, PTolmm

Examples

```
## Not run:
    # School Girls Data Example

      data(schoolgirls)
      attach(schoolgirls)

    # Prior information

      prior<-list(alpha=1,nu0=4.01,tau1=0.01,tau2=0.01,
                  tinv=diag(10,2),mub=rep(0,2),Sb=diag(1000,2))

    # Initial state
      state <- NULL

    # MCMC parameters

      nburn<-5000
      nsave<-10000
      nskip<-20
      ndisplay<-1000
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay)

    # Fit the model: First run

      fit1<-DPlmm(fixed=height~1,random=~age|child,prior=prior,mcmc=mcmc,
                  state=state,status=TRUE)
      fit1
    # Fit the model: Continuation
      state<-fit1$state

      fit2<-DPlmm(fixed=height~1,random=~age|child,prior=prior,mcmc=mcmc,
                  state=state,status=FALSE)
      fit2

    # Summary with HPD and Credibility intervals
      summary(fit2)
      summary(fit2,hpd=FALSE)


    # Plot model parameters
    # (to see the plots gradually set ask=TRUE)
      plot(fit2,ask=FALSE)
      plot(fit2,ask=FALSE,nfigr=2,nfigc=2)

    # Plot an specific model parameter
    # (to see the plots gradually set ask=TRUE)
      plot(fit2,ask=FALSE,nfigr=1,nfigc=2,param="sigma-(Intercept)")
      plot(fit2,ask=FALSE,nfigr=1,nfigc=2,param="ncluster")

## End(Not run)
```

DPMdencens *Bayesian density estimation for interval-censored data using a DPM of normals*

Description

This function generates a posterior density sample for a Dirichlet process mixture of normals model for interval-censored data.

Usage

```
DPMdencens(left,right,ngrid=100,grid=NULL,prior,mcmc,state,status)
```

Arguments

`left`	a vector or matrix giving the lower limit for each response variable. Note that the responses are defined on the entire real line and that unknown limits should be indicated by NA.
`right`	a vector or matrix giving the upper limit for each response variable. Note that the responses are defined on the entire real line and that unknown limits should be indicated by NA.
`ngrid`	number of grid points where the density estimate is evaluated. The default value is 100.

`grid`	matrix of dimension ngrid*nvar of grid points where the density estimate is evaluated. The default value is NULL and the grid is chosen according to the range of the interval limits.
`prior`	a list giving the prior information. The list includes the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing, see details below), `nu2` and `psiinv2` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix, `Psi1`, of the inverted Wishart part of the baseline distribution, `tau1` and `tau2` giving the hyperparameters for the gamma prior distribution of the scale parameter `k0` of the normal part of the baseline distribution, `m2` and `s2` giving the mean and the covariance of the normal prior for the mean, `m1`, of the normal component of the baseline distribution, respectively, `nu1` and `psiinv1` (it must be specified if `nu2` is missing, see details below) giving the hyperparameters of the inverted Wishart part of the baseline distribution and, `m1` giving the mean of the normal part of the baseline distribution (it must be specified if `m2` is missing, see details below) and, `k0` giving the scale parameter of the normal part of the baseline distribution (it must be specified if `tau1` is missing, see details below).
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.

Details

This generic function fits a Dirichlet process mixture of normal model for density estimation (Escobar and West, 1995) based on interval-censored data:

$$y_{ij} \in [l_{ij}, u_{ij}), i = 1, \ldots, n, j = 1, \ldots, m$$

$$y_i | \mu_i, \Sigma_i \sim N(\mu_i, \Sigma_i), i = 1, \ldots, n,$$

$$(\mu_i, \Sigma_i) | G \sim G,$$

$$G | \alpha, G_0 \sim DP(\alpha G_0),$$

where, $y_i = (y_{i1}, \ldots, y_{im})$, and the baseline distribution is the conjugate normal-inverted-Wishart distribution,

$$G_0 = N(\mu | m_1, (1/k_0)\Sigma) IW(\Sigma | \nu_1, \psi_1)$$

To complete the model specification, independent hyperpriors are assumed (optional),

$$\alpha | a_0, b_0 \sim Gamma(a_0, b_0)$$

$$m_1|m_2, s_2 \sim N(m_2, s_2)$$
$$k_0|\tau_1, \tau_2 \sim Gamma(\tau_1/2, \tau_2/2)$$
$$\psi_1|\nu_2, \psi_2 \sim IW(\nu_2, \psi_2)$$

Note that the inverted-Wishart prior is parametrized such that if $A \sim IW_q(\nu, \psi)$ then $E(A) = \psi^{-1}/(\nu - q - 1)$.

To let part of the baseline distribution fixed at a particular value, set the corresponding hyperparameters of the prior distributions to NULL in the hyperprior specification of the model.

Although the baseline distribution, G_0, is a conjugate prior in this model specification, an algorithm based on auxiliary parameters is adopted. Specifically, the algorithm 8 with $m = 1$ of Neal (2000) is considered in the `DPMdencens` function.

Finally, note that this function can be used to fit the DPM of normals model for ordinal data proposed by Kottas, Mueller and Quintana (2005). In this case, the arbitrary cut-off points must be specified in `left` and `right`. Samples from the predictive distribution contained in the (last columns) of the object randsave (please see below) can be used to obtain an estimate of the cell probabilities.

Value

An object of class `DPMdencens` representing the DP mixture of normals model fit. Generic functions such as `print`, `summary`, and `plot` have methods to show the results of the fit. The results include the baseline parameters, `alpha`, and the number of clusters.

The function `DPrandom` can be used to extract the posterior mean of the subject-specific means and covariance matrices.

The MCMC samples of the parameters and the errors in the model are stored in the object `thetasave` and `randsave`, respectively. Both objects are included in the list `save.state` and are matrices which can be analyzed directly by functions provided by the coda package.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`ncluster`	an integer giving the number of clusters.
`muclus`	a matrix of dimension (nobservations+100)*(nvariables) giving the means of the clusters (only the first `ncluster` are considered to start the chain).
`sigmaclus`	a matrix of dimension (nobservations+100)*((nvariables)*((nvariables)+1)/2) giving the lower matrix of the covariance matrix of the clusters (only the first `ncluster` are considered to start the chain).
`ss`	an interger vector defining to which of the `ncluster` clusters each observation belongs.
`alpha`	giving the value of the precision parameter.
`m1`	giving the mean of the normal components of the baseline distribution.
`k0`	giving the scale parameter of the normal part of the baseline distribution.
`psi1`	giving the scale matrix of the inverted-Wishart part of the baseline distribution.
`y`	giving the matrix of imputed data points.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

References

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Kottas, A., Mueller, P., Quintana, F. (2005). Nonparametric Bayesian modeling for multivariate ordinal data. Journal of Computational and Graphical Statistics, 14: 610-625.

Neal, R. M. (2000). Markov Chain sampling methods for Dirichlet process mixture models. Journal of Computational and Graphical Statistics, 9: 249-265.

See Also

`DPrandom`, `DPdensity`

Examples

```
## Not run:
    ####################################
    # Bivariate example:
    # Censored data is artificially
    # created
    ####################################

      data(airquality)
      attach(airquality)

      ozone <- Ozone**(1/3)
      radiation <- Solar.R
      y <- na.omit(cbind(radiation,ozone))

    # create censored-data
      xxlim <- seq(0,300,50)
      yylim <- seq(1.5,5.5,1)

      left <- matrix(0,nrow=nrow(y),ncol=2)
      right <- matrix(0,nrow=nrow(y),ncol=2)

      for(i in 1:nrow(y))
      {
          left[i,1] <- NA
          right[i,1] <- NA
          if(y[i,1] < xxlim[1]) right[i,1] <- xxlim[1]
          for(j in 1:length(xxlim))
          {
              if(y[i,1] >= xxlim[j]) left[i,1] <- xxlim[j]
              if(y[i,1] >= xxlim[j]) right[i,1] <- xxlim[j+1]
          }
          left[i,2] <- NA
          right[i,2] <- NA
```

```
        if(y[i,2] < yylim[1]) right[i,2] <- yylim[1]

        for(j in 1:length(yylim))
        {
            if(y[i,2] >= yylim[j]) left[i,2] <- yylim[j]
            if(y[i,2] >= yylim[j]) right[i,2] <- yylim[j+1]
        }
    }

# Prior information
  s2 <- matrix(c(10000,0,0,1),ncol=2)
  m2 <- c(180,3)
  psiinv2 <- diag(c(1/10000,1),2)

  prior <- list(alpha=1,nu1=4,nu2=4,s2=s2,
                m2=m2,psiinv2=psiinv2,tau1=0.01,tau2=0.01)

# Initial state
  state <- NULL

# MCMC parameters
  nburn <- 5000
  nsave <- 5000
  nskip <- 3
  ndisplay <- 1000
  mcmc <- list(nburn=nburn,
               nsave=nsave,
               nskip=nskip,
               ndisplay=ndisplay)

# Fitting the model
  fit1 <- DPMdencens(left=left,right=right,ngrid=100,
                     prior=prior,mcmc=mcmc,
                     state=state,status=TRUE)
  fit1
  summary(fit1)

# Plot the estimated density
  plot(fit1)

# Extracting the univariate density estimates
  cbind(fit1$grid[,1],fit1$funi[[1]])
  cbind(fit1$grid[,2],fit1$funi[[2]])

# Extracting the bivariate density estimates
  fit1$grid[,1]
  fit1$grid[,2]
  fit1$fbiv[[1]]

# Plot of the estimated density along with the
# true data points and censoring limits
  contour(fit1$grid[,1],fit1$grid[,2],fit1$fbiv[[1]])
  points(y)
```

```
    abline(v=xxlim)
    abline(h=yylim)

## End(Not run)
```

DPmeta *Bayesian analysis for a semiparametric linear mixed effects meta-analysis model using a MDP*

Description

This function generates a posterior density sample for a semiparametric linear mixed effects meta-analysis model using a Dirichlet process or a Mixture of Dirichlet process prior for the distribution of the random effects.

Usage

```
DPmeta(formula,prior,mcmc,state,status,data=sys.frame(sys.parent()),
       na.action=na.fail)
```

Arguments

`formula`	a two-sided linear formula object describing the fixed-effects part of the model, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right. Both effect and variance must be included in the LHS of the formula object
`prior`	a list giving the prior information. The list include the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` and `b0` are missing, see details below), `tau1` and `tau2` giving the hyperparameters for the prior distribution of the variance of the centering distribution, `sigma2` giving the value of the variance of the centering distribution (it must be specified if `tau1` and `tau2` are missing), `mub` and `Sb` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `mu` giving the value of the mean of the centering distribution (it must be specified if `mub` and `Sb` are missing), and `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the fixed effects (must be specified only if fixed effects are considered in the model).
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.

`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `DPmeta` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a semiparametric linear mixed effects meta-analysis model:

$$y_i \sim N(\theta_i + X_i\beta, \sigma^2_{ei}), i = 1, \ldots, n$$

$$\theta_i | G \sim G$$

$$G|\alpha, G_0 \sim DP(\alpha G_0)$$

where, $G_0 = N(\theta|\mu, \sigma^2)$. To complete the model specification, independent hyperpriors are assumed,

$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\beta|\beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

$$\mu|\mu_b, S_b \sim N(\mu_b, S_b)$$

$$\sigma^{-2}|\tau_1, \tau_2 \sim Gamma(\tau_1/2, \tau_2/2)$$

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

The computational implementation of the model is based on the marginalization of the `DP` and on the use of MCMC methods for conjugate priors (Escobar, 1994; Escobar and West, 1998).

The average effect is sampled using the ϵ-DP approximation proposed by Muliere and Tardella (1998), with ϵ=0.01.

Value

An object of class `DPmeta` representing the linear mixed-effects model fit. Generic functions such as `print`, `plot`, `summary`, and `anova` have methods to show the results of the fit. The results include `beta`, `mu`, `sigma2`, `alpha`, and the number of clusters.

The function `DPrandom` can be used to extract the posterior mean of the random effects.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`ncluster`	an integer giving the number of clusters.

`alpha`	giving the value of the precision parameter
`b`	a vector of dimension (nsubjects) giving the value of the random effects for each subject.
`bclus`	a vector of dimension (nsubjects) giving the value of the random effects for each clusters (only the first `ncluster` are considered to start the chain).
`ss`	an interger vector defining to which of the `ncluster` clusters each subject belongs.
`beta`	giving the value of the fixed effects.
`mu`	giving the mean of the normal baseline distributions.
`sigma2`	giving the variance of the normal baseline distributions.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

References

Escobar, M.D. (1994) Estimating Normal Means with a Dirichlet Process Prior, Journal of the American Statistical Association, 89: 268-277.

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Escobar, M.D. and West, M. (1998) Computing Bayesian Nonparametric Hierarchical Models, in Practical Nonparametric and Semiparametric Bayesian Statistics, eds: D. Dey, P. Muller, D. Sinha, New York: Springer-Verlag, pp. 1-22.

Muliere, P. and Tardella, L. (1998) Approximating distributions of random functionals of Ferguson-Dirichlet priors. The Canadian Journal of Statistics, 26(2): 283-297.

See Also

`DPrandom`, `DPMmeta`, `DPlmm` , `DPglmm` , `DPolmm` , `DPMlmm` , `DPMglmm`, `DPMolmm`

Examples

```
## Not run:

    ##################################################################
    # Data on the effectiveness of silver sulfadiazine coating
    # on venous catheters for preventing bacterial colonisation of
    # the catheter and bloodstream infection.
    # Veenstra D et al (1998) "Efficacy of Antiseptic Impregnated
    # Central Venous Catheters in Preventing Nosocomial Infections:
    # A Meta-analysis" JAMA 281:261-267.
    #
    # Note that -Inf and Inf have been replaced by NA.
    ##################################################################

      studies <- c("Tennenberg","Maki","vanHeerden",
                   "Hannan","Bach(a)","Bach(b)",
                    "Heard","Collins","Ciresi","Ramsay",
                    "Trazzera","George")

      logOR <- c(-1.5187189,-0.7136877,-1.3217558,-0.1910552,
                  NA,-2.2005195,-0.5057461,-2.3538784,-0.3643810,
                  -0.5371429,-0.7608058,-2.1400662)

      varlogOR <- c(0.4157541,0.2632550,0.6739189,0.3727788,NA,
                    0.7623470,0.2306169,0.7477891,0.3645463,0.2291839,
                    0.3561542,0.5190489)^2

      names(logOR) <- studies
      names(varlogOR) <- studies
      y <- cbind(logOR,varlogOR)
      colnames(y) <- c("logOR","varlogOR")

    # Prior information

      prior<-list(alpha=1,
                  tau1=20,
                  tau2=10,
                  mub=0,
                  Sb=100)

    # Initial state
      state <- NULL


    # MCMC parameters

      nburn<-20000
      nsave<-10000
      nskip<-20
      ndisplay<-100
      mcmc <- list(nburn=nburn,
                   nsave=nsave,
                   nskip=nskip,
                   ndisplay=ndisplay)

    # Fit the model: First run

      fit1<-DPmeta(formula=y~1,prior=prior,mcmc=mcmc,
                   state=state,status=TRUE)
      fit1

    # Summary with HPD and Credibility intervals
      summary(fit1)
      summary(fit1,hpd=FALSE)

    # Plot model parameters (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE)
      plot(fit1,ask=FALSE,nfigr=2,nfigc=2)
```

```
## End(Not run)
```

`DPMglmm` *Bayesian analysis for a semiparametric generalized linear mixed model using a DPM of normals*

Description

This function generates a posterior density sample for a semiparametric generalized linear mixed model using a Dirichlet Process Mixture of Normals prior for the distribution of the random effects.

Usage

```
DPMglmm(fixed,random,family,offset,n,prior,mcmc,state,status,
      data=sys.frame(sys.parent()),na.action=na.fail)
```

Arguments

`fixed`	a two-sided linear formula object describing the fixed-effects part of the model, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right.
`random`	a one-sided formula of the form `~z1+...+zn | g`, with `z1+...+zn` specifying the model for the random effects and `g` the grouping variable. The random effects formula will be repeated for all levels of grouping.
`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. The families(links) considered by `DPglmm` so far are binomial(logit), binomial(probit), Gamma(log), and poisson(log). The gaussian(identity) case is implemented separately in the function `DPlmm`.
`offset`	this can be used to specify an a priori known component to be included in the linear predictor during the fitting (only for poisson and gamma models).
`n`	this can be used to indicate the total number of cases in a binomial model (only implemented for the logistic link). If it is not specified the response variable must be binary.
`prior`	a list giving the prior information. The list include the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` and `b0` are missing, see details below), `nu0` and `Tinv` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix of the normal kernel, `mb` and `Sb` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution,`nub` and `Tbinv` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix of the normal baseline distribution, `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the fixed

	effects (must be specified only if fixed effects are considered in the model) and, `tau1` and `tau2` giving the hyperparameters for the prior distribution for the inverse of the dispersion parameter of the Gamma model (they must be specified only if the Gamma model is considered).
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, `ndisplay` giving the number of saved scans to be displayed on the screen (the function reports on the screen when every `ndisplay` iterations have been carried out), `tune1` giving the positive Metropolis tuning parameter for the precision parameter of the Gamma model (the default value is 1.1).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `DPMglmm` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a generalized linear mixed-effects model, where the linear predictor is modeled as follows:

$$\eta_i = X_i\beta_F + Z_i\beta_R + Z_i b_i, i = 1, \ldots, n$$

$$\theta_i | G, \Sigma \sim \int N(\mu, \Sigma) G(d\mu)$$

$$G | \alpha, \mu_b, \Sigma_b \sim DP(\alpha N(\mu_b, \Sigma_b))$$

where, $\theta_i = \beta_R + b_i$, $\beta = \beta_F$, and $G_0 = N(\theta|\mu, \Sigma)$. To complete the model specification, independent hyperpriors are assumed,

$$\beta | \beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

$$\Sigma | \nu_0, T \sim IW(\nu_0, T)$$

$$\alpha | a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\mu_b | m_b, S_b \sim N(m_b, S_b)$$

$$\Sigma_b | \nu_b, Tb \sim IW(\nu_b, Tb)$$

Note that the inverted-Wishart prior is parametrized such that $E(\Sigma) = T^{-1}/(\nu_0 - q - 1)$.

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value set, a_0 to NULL in the prior specification.

The inverse of the dispersion parameter of the Gamma model is modeled using gamma distribution, $\Gamma(\tau_1/2, \tau_2/2)$.

The computational implementation of the model is based on the marginalization of the DP and the MCMC is model-specific.

For the poisson, Gamma, and binomial(logit), MCMC methods for nonconjugate priors (see, MacEachern and Muller, 1998; Neal, 2000) are used. Specifically, the algorithm 8 with m=1 of Neal (2000), is considered in the DPMglmm function. In this case, the fully conditional distributions for fixed and in the resampling step of random effects are generated through the Metropolis-Hastings algorithm with a IWLS proposal (see, West, 1985 and Gamerman, 1997).

For the binomial(probit) the following latent variable representation is used:

$$y_{ij} = I(w_{ij} > 0), j = 1, \ldots, n_i$$

$$w_{ij} | \beta, \theta_i, \lambda_i \sim N(X_{ij}\beta + Z_{ij}\theta_i, 1)$$

In this case, the computational implementation of the model is based on the marginalization of the DP and on the use of MCMC methods for conjugate priors for a collapsed state described by MacEachern (1998).

The β_R parameters are sampled using the ϵ-DP approximation proposed by Muliere and Tardella (1998), with ϵ=0.01.

Value

An object of class DPMglmm representing the generalized linear mixed-effects model fit. Generic functions such as print, plot, summary, and anova have methods to show the results of the fit. The results include betaR, betaF, sigma2e, Sigma, mub, the elements of Sigmab, alpha, and the number of clusters.

The function DPMrandom can be used to extract the posterior mean of the random effects.

The list state in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set status=TRUE and create the list state based on this starting values. In this case the list state must include the following objects:

ncluster	an integer giving the number of clusters.
alpha	giving the value of the precision parameter
b	a matrix of dimension (nsubjects)*(nrandom effects) giving the value of the random effects for each subject.
mu	a matrix of dimension (nsubjects)*(nrandom effects) giving the value of the means of the normal kernel for each cluster (only the first ncluster are considered to start the chain).
ss	an interger vector defining to which of the ncluster clusters each subject belongs.
beta	giving the value of the fixed effects.
sigma	giving the variance matrix of the normal kernel.
mub	giving the mean of the normal baseline distributions.
sigmab	giving the variance matrix of the normal baseline distributions.
phi	giving the dispersion parameter for the Gamma model (if needed).

Author(s)

Alejandro Jara <<atjara@uc.cl>>

References

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Gamerman, D. (1997) Sampling from the posterior distribution in generalized linear mixed models. Statistics and Computing, 7: 57-68.

MacEachern, S.N. (1998) Computational Methods for Mixture of Dirichlet Process Models, in Practical Nonparametric and Semiparametric Bayesian Statistics, eds: D. Dey, P. Muller, D. Sinha, New York: Springer-Verlag, pp. 1-22.

MacEachern, S. N. and Muller, P. (1998) Estimating mixture of Dirichlet Process Models. Journal of Computational and Graphical Statistics, 7 (2): 223-338.

Muliere, P. and Tardella, L. (1998) Approximating distributions of random functionals of Ferguson-Dirichlet priors. The Canadian Journal of Statistics, 26(2): 283-297.

Neal, R. M. (2000) Markov Chain sampling methods for Dirichlet process mixture models. Journal of Computational and Graphical Statistics, 9:249-265.

West, M. (1985) Generalized linear models: outlier accomodation, scale parameter and prior distributions. In Bayesian Statistics 2 (eds Bernardo et al.), 531-558, Amsterdam: North Holland.

See Also

DPMrandom, DPMlmm , DPMolmm, DPlmm , DPglmm , DPolmm, PTlmm , PTglmm, PTolmm

Examples

```
## Not run:
    # Respiratory Data Example

      data(indon)
      attach(indon)

      baseage2<-baseage**2
      follow<-age-baseage
      follow2<-follow**2

    # Prior information

      prior<-list(alpha=1,
                  nu0=4.01,
                  tinv=diag(1,1),
                  nub=4.01,
                  tbinv=diag(1,1),
                  mb=rep(0,1),
                  Sb=diag(1000,1),
                  beta0=rep(0,9),
                  Sbeta0=diag(1000,9))
    # Initial state
      state <- NULL

    # MCMC parameters

      nburn<-5000
      nsave<-25000
      nskip<-20
      ndisplay<-1000
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay)

    # Fit the Probit model
      fit1<-DPMglmm(fixed=infect~gender+height+cosv+sinv+xero+baseage+
                    baseage2+follow+follow2,
                    random=~1|id,family=binomial(probit),
                    prior=prior,mcmc=mcmc,state=state,status=TRUE)

    # Fit the Logit model
      fit2<-DPMglmm(fixed=infect~gender+height+cosv+sinv+xero+baseage+
                    baseage2+follow+follow2,random=~1|id,
                    family=binomial(logit),
                    prior=prior,mcmc=mcmc,state=state,status=TRUE)

    # Summary with HPD and Credibility intervals
      summary(fit1)
      summary(fit1,hpd=FALSE)

      summary(fit2)
      summary(fit2,hpd=FALSE)


    # Plot model parameters (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE)
      plot(fit1,ask=FALSE,nfigr=2,nfigc=2)

    # Plot an specific model parameter (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE,nfigr=1,nfigc=2,param="baseage")
      plot(fit1,ask=FALSE,nfigr=1,nfigc=2,param="ncluster")

## End(Not run)
```

`DPMlmm` *Bayesian analysis for a semiparametric linear mixed model using a DPM of normals*

Description

This function generates a posterior density sample for a semiparametric linear mixed model using a Dirichlet Process Mixture of Normals prior for the distribution of the random effects.

Usage

```
DPMlmm(fixed,random,prior,mcmc,state,status,
       data=sys.frame(sys.parent()),
       na.action=na.fail)
```

Arguments

`fixed`	a two-sided linear formula object describing the fixed-effects part of the model, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right.
`random`	a one-sided formula of the form `~z1+...+zn | g`, with `z1+...+zn` specifying the model for the random effects and `g` the grouping variable. The random effects formula will be repeated for all levels of grouping.
`prior`	a list giving the prior information. The list include the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` and `b0` are missing, see details below), `nu0` and `Tinv` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix of the normal kernel, `mb` and `Sb` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution,`nub` and `Tbinv` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix of the normal baseline distribution, `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the fixed effects (must be specified only if fixed effects are considered in the model) and, `tau1` and `tau2` giving the hyperparameters for the prior distribution of the error variance.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `DPMlmm` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a linear mixed-effects model (Verbeke and Molenberghs, 2000):

$$y_i \sim N(X_i\beta_F + Z_i\beta_R + Z_ib_i, \sigma_e^2 I_{n_i}), i = 1, \ldots, n$$

$$\theta_i|G, \Sigma \sim \int N(\mu, \Sigma)G(d\mu)$$
$$G|\alpha, \mu_b, \Sigma_b \sim DP(\alpha N(\mu_b, \Sigma_b))$$
$$\sigma_e^{-2}|\tau_1, \tau_2 \sim \Gamma(\tau_1/2, \tau_2/2)$$

where, $\theta_i = \beta_R + b_i$, $\beta = \beta_F$, and $G_0 = N(\theta|\mu, \Sigma)$. To complete the model specification, independent hyperpriors are assumed,

$$\beta|\beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$
$$\Sigma|\nu_0, T \sim IW(\nu_0, T)$$
$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$
$$\mu_b|m_b, S_b \sim N(m_b, S_b)$$
$$\Sigma_b|\nu_b, Tb \sim IW(\nu_b, Tb)$$

Note that the inverted-Wishart prior is parametrized such that $E(\Sigma) = T^{-1}/(\nu_0 - q - 1)$.

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

The computational implementation of the model is based on the marginalization of the `DP` and on the use of MCMC methods for conjugate priors for a collapsed state of MacEachern (1998).

The β_R parameters are sampled using the ϵ-DP approximation proposed by Muliere and Tardella (1998), with ϵ=0.01.

Value

An object of class `DPMlmm` representing the linear mixed-effects model fit. Generic functions such as `print`, `plot`, `summary`, and `anova` have methods to show the results of the fit. The results include `betaR`, `betaF`, `sigma2e`, `Sigma`, `mub`, the elements of `Sigmab`, `alpha`, and the number of clusters.

The function `DPrandom` can be used to extract the posterior mean of the random effects.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`ncluster`	an integer giving the number of clusters.
`alpha`	giving the value of the precision parameter
`b`	a matrix of dimension (nsubjects)*(nrandom effects) giving the value of the random effects for each subject.
`mu`	a matrix of dimension (nsubjects)*(nrandom effects) giving the value of the means of the normal kernel for each cluster (only the first `ncluster` are considered to start the chain).
`ss`	an interger vector defining to which of the `ncluster` clusters each subject belongs.

beta	giving the value of the fixed effects.
sigma	giving the variance matrix of the normal kernel.
mub	giving the mean of the normal baseline distributions.
sigmab	giving the variance matrix of the normal baseline distributions.
sigma2e	giving the error variance.

Author(s)

Alejandro Jara <<atjara@uc.cl>>

References

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

MacEachern, S.N. (1998) Computational Methods for Mixture of Dirichlet Process Models, in Practical Nonparametric and Semiparametric Bayesian Statistics, eds: D. Dey, P. Muller, D. Sinha, New York: Springer-Verlag, pp. 1-22.

Muliere, P. and Tardella, L. (1998) Approximating distributions of random functionals of Ferguson-Dirichlet priors. The Canadian Journal of Statistics, 26(2): 283-297.

Verbeke, G. and Molenberghs, G. (2000). Linear mixed models for longitudinal data, New York: Springer-Verlag.

See Also

DPMrandom, DPMglmm, DPMolmm, DPlmm , DPglmm, DPolmm, PTlmm , PTglmm, PTolmm

Examples

```
## Not run:
    # School Girls Data Example

      data(schoolgirls)
      attach(schoolgirls)

    # Prior information

      prior<-list(alpha=1,
                  tau1=0.01,tau2=0.01,
                  nu0=4.01,
                  tinv=diag(10,2),
                  nub=4.01,
                  tbinv=diag(10,2),
                  mb=rep(0,2),
                  Sb=diag(1000,2))

    # Initial state
      state <- NULL

    # MCMC parameters
      nburn<-5000
      nsave<-10000
      nskip<-20
      ndisplay<-1000
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay)

    # Fit the model: First run

      fit1<-DPMlmm(fixed=height~1,random=~age|child,prior=prior,mcmc=mcmc,
                   state=state,status=TRUE)
      fit1

    # Fit the model: Continuation
      state<-fit1$state

      fit2<-DPMlmm(fixed=height~1,random=~age|child,prior=prior,mcmc=mcmc,
                  state=state,status=FALSE)
      fit2

    # Summary with HPD and Credibility intervals
      summary(fit2)
      summary(fit2,hpd=FALSE)


    # Plot model parameters
    # (to see the plots gradually set ask=TRUE)
      plot(fit2,ask=FALSE)
      plot(fit2,ask=FALSE,nfigr=2,nfigc=2)

    # Plot an specific model parameter
    # (to see the plots gradually set ask=TRUE)
      plot(fit2,ask=FALSE,nfigr=1,nfigc=2,param="sigma-(Intercept)")
      plot(fit2,ask=FALSE,nfigr=1,nfigc=2,param="ncluster")

## End(Not run)
```

DPMmeta — *Bayesian analysis for a semiparametric linear mixed effects meta-analysis model using a DPM of normals*

Description

This function generates a posterior density sample for a semiparametric linear mixed effects meta-analysis model using a Dirichlet Process Mixture of Normals prior for the distribution of the random effects.

Usage

```
DPMmeta(formula,prior,mcmc,state,status,
        data=sys.frame(sys.parent()),
        na.action=na.fail)
```

Arguments

`formula`	a two-sided linear formula object describing the fixed-effects part of the model, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right.Both effect and variance must be included in the LHS of the formula object
`prior`	a list giving the prior information. The list include the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` and `b0` are missing, see details below), `tau01` and `tau02` giving the hyperparameters of the inverted gamma prior distribution for the variance of the normal kernel, `mb` and `Sb` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `mub` giving the value of the mean of the centering distribution (it must be specified if `mb` and `Sb` are missing), `tau11` and `tau12` giving the hyperparameters of the inverted gamma prior distribution for the variance of the normal baseline distribution, `sigmab` giving the value of the variance of the centering distribution (it must be specified if `tau11` and `tau12` are missing), and `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the fixed effects (must be specified only if fixed effects are considered in the model).
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `DPMmeta` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a semiparametric linear mixed effects meta-analysis model:

$$y_i \sim N(\theta_i + X_i\beta, \sigma^2_{ei}), i = 1, \ldots, n$$

$$\theta_i | G, \sigma^2 \sim \int N(\mu, \sigma^2) G(d\mu)$$

$$\sigma^{-2}|\tau_{01}, \tau_{02} \sim Gamma(\tau_{01}/2, \tau_{02}/2)$$

$$G|\alpha, G_0 \sim DP(\alpha G_0)$$

where, $G_0 = N(\mu|\mu_b, \sigma_b^2)$. To complete the model specification, independent hyperpriors are assumed,

$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\beta|\beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

$$\mu_b|m_b, S_b \sim N(m_b, S_b)$$

$$\sigma_b^{-2}|\tau_{11}, \tau_{12} \sim Gamma(\tau_{11}/2, \tau_{12}/2)$$

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

The computational implementation of the model is based on the marginalization of the `DP` and on the use of MCMC methods for conjugate priors for a collapsed state of MacEachern (1998).

The average effect is sampled using the ϵ-DP approximation proposed by Muliere and Tardella (1998), with ϵ=0.01.

Value

An object of class `DPMmeta` representing the linear mixed-effects model fit. Generic functions such as `print`, `plot`, `summary`, and `anova` have methods to show the results of the fit. The results include `beta`, `sigma2`, `mub`, `sigma2b`, `alpha`, and the number of clusters.

The function `DPMrandom` can be used to extract the posterior mean of the random effects.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`ncluster`	an integer giving the number of clusters.
`alpha`	giving the value of the precision parameter
`b`	a vector of dimension (nrec) giving the value of the random effects for each subject.
`mu`	a vector of dimension (nrec) giving the value of the mean of the normal kernel for each cluster (only the first `ncluster` are considered to start the chain).
`ss`	an interger vector defining to which of the `ncluster` clusters each subject belongs.
`beta`	giving the value of the fixed effects.
`sigma2`	giving the variance of the normal kernel.
`mub`	giving the mean of the normal baseline distributions.
`sigma2b`	giving the variance of the normal baseline distributions.

Author(s)

Alejandro Jara <<atjara@uc.cl>>

References

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

MacEachern, S.N. (1998) Computational Methods for Mixture of Dirichlet Process Models, in Practical Nonparametric and Semiparametric Bayesian Statistics, eds: D. Dey, P. Muller, D. Sinha, New York: Springer-Verlag, pp. 23-44.

Muliere, P. and Tardella, L. (1998) Approximating distributions of random functionals of Ferguson-Dirichlet priors. The Canadian Journal of Statistics, 26(2): 283-297.

See Also

DPMrandom, DPmeta DPMglmm, DPMolmm, DPlmm , DPglmm, DPolmm

Examples

```
## Not run:

    ###################################################################
    # Data on the effectiveness of silver sulfadiazine coating
    # on venous catheters for preventing bacterial colonisation of
    # the catheter and bloodstream infection.
    # Veenstra D et al (1998) "Efficacy of Antiseptic Impregnated
    # Central Venous Catheters in Preventing Nosocomial Infections:
    # A Meta-analysis" JAMA 281:261-267.
    #
    # Note that -Inf and Inf have been replaced by NA.
    ###################################################################

      studies <- c("Tennenberg","Maki","vanHeerden",
                   "Hannan","Bach(a)","Bach(b)",
                   "Heard","Collins","Ciresi","Ramsay",
                   "Trazzera","George")

      logOR <- c(-1.5187189,-0.7136877,-1.3217558,-0.1910552,
                 NA,-2.2005195,-0.5057461,-2.3538784,-0.3643810,
                 -0.5371429,-0.7608058,-2.1400662)

      varlogOR <- c(0.4157541,0.2632550,0.6739189,0.3727788,NA,
                    0.7623470,0.2306169,0.7477891,0.3645463,0.2291839,
                    0.3561542,0.5190489)^2

      names(logOR) <- studies
      names(varlogOR) <- studies
      y <- cbind(logOR,varlogOR)
      colnames(y) <- c("logOR","varlogOR")

    # Prior information
    prior<-list(alpha=1,
                tau01=20,
                tau02=10,
                tau11=20,
                tau12=10,
                mb=0,
                Sb=100)

  # Initial state
    state <- NULL


  # MCMC parameters

    nburn<-20000
    nsave<-10000
    nskip<-20
    ndisplay<-100
    mcmc <- list(nburn=nburn,
                 nsave=nsave,
                 nskip=nskip,
                 ndisplay=ndisplay)

  # Fit the model: First run

    fit1<-DPMmeta(formula=y~1,prior=prior,mcmc=mcmc,
                  state=state,status=TRUE)
    fit1

  # Summary with HPD and Credibility intervals
    summary(fit1)
    summary(fit1,hpd=FALSE)

  # Plot model parameters (to see the plots gradually set ask=TRUE)
    plot(fit1,ask=FALSE)
    plot(fit1,ask=FALSE,nfigr=2,nfigc=2)

## End(Not run)
```

`DPMolmm` *Bayesian analysis for a semiparametric ordinal linear mixed model using a DPM of normals*

Description

This function generates a posterior density sample for a semiparametric ordinal linear mixed model using a Dirichlet Process Mixture of Normals prior for the distribution of the random effects.

Usage

```
DPMolmm(fixed,random,prior,mcmc,state,status,
        data=sys.frame(sys.parent()),
        na.action=na.fail)
```

Arguments

`fixed`	a two-sided linear formula object describing the fixed-effects part of the model, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right.
`random`	a one-sided formula of the form `~z1+...+zn | g`, with `z1+...+zn` specifying the model for the random effects and `g` the grouping variable. The random effects formula will be repeated for all levels of grouping.
`prior`	a list giving the prior information. The list include the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` and `b0` are missing, see details below), `nu0` and `Tinv` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix of the normal kernel, `mb` and `Sb` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution,`nub` and `Tbinv` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix of the normal baseline distribution, and `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the fixed effects (must be specified only if fixed effects are considered in the model).
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `DPMolmm` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits an ordinal linear mixed-effects model with a probit link (see, e.g., Molenberghs and Verbeke, 2005):

$$Y_{ij} = k, \text{ if } \gamma_{k-1} \leq W_{ij} < \gamma_k, k = 1, \ldots, K$$

$$W_{ij} \mid \beta_F, \beta_R, b_i \sim N(X_{ij}\beta_F + Z_{ij}\beta_R + Z_{ij}b_i, 1), i = 1, \ldots, N, j = 1, \ldots, n_i$$

$$\theta_i | G, \Sigma \sim \int N(m, \Sigma) G(dm)$$

$$G | \alpha, \mu_b, \Sigma_b \sim DP(\alpha N(\mu_b, \Sigma_b))$$

where, $\theta_i = \beta_R + b_i$, $\beta = \beta_F$, and $G_0 = N(\theta|\mu, \Sigma)$. To complete the model specification, independent hyperpriors are assumed,

$$\beta | \beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

$$\Sigma | \nu_0, T \sim IW(\nu_0, T)$$

$$\alpha | a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\mu_b | m_b, S_b \sim N(m_b, S_b)$$

$$\Sigma_b | \nu_b, Tb \sim IW(\nu_b, Tb)$$

A uniform prior is used for the cutoff points. Note that the inverted-Wishart prior is parametrized such that $E(\Sigma) = T^{-1}/(\nu_0 - q - 1)$.

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

The computational implementation of the model is based on the marginalization of the `DP` and on the use of MCMC methods for conjugate priors for a collapsed state of MacEachern (1998).

The β_R parameters are sampled using the ϵ-DP approximation proposed by Muliere and Tardella (1998), with ϵ=0.01.

Value

An object of class `DPMolmm` representing the linear mixed-effects model fit. Generic functions such as `print`, `plot`, `summary`, and `anova` have methods to show the results of the fit. The results include `betaR`, `betaF`, `mu`, the elements of `Sigma`, `mub`, the elements of `Sigmab`, the cutoff points, `alpha`, and the number of clusters.

The function `DPMrandom` can be used to extract the posterior mean of the random effects.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`ncluster`	an integer giving the number of clusters.
`alpha`	giving the value of the precision parameter
`b`	a matrix of dimension (nsubjects)*(nrandom effects) giving the value of the random effects for each subject.
`cutoff`	a real vector defining the cutoff points. Note that the first cutoff must be fixed to 0 in this function.

mu	a matrix of dimension (nsubjects)*(nrandom effects) giving the value of the means of the normal kernel for each cluster (only the first ncluster are considered to start the chain).
ss	an interger vector defining to which of the ncluster clusters each subject belongs.
beta	giving the value of the fixed effects.
sigma	giving the variance matrix of the normal kernel.
mub	giving the mean of the normal baseline distributions.
sigmab	giving the variance matrix of the normal baseline distributions.

Author(s)

Alejandro Jara <<atjara@uc.cl>>

References

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

MacEachern, S.N. (1998) Computational Methods for Mixture of Dirichlet Process Models, in Practical Nonparametric and Semiparametric Bayesian Statistics, eds: D. Dey, P. Muller, D. Sinha, New York: Springer-Verlag, pp. 1-22.

Molenberghs, G. and Verbeke, G. (2005). Models for discrete longitudinal data, New York: Springer-Verlag.

Muliere, P. and Tardella, L. (1998) Approximating distributions of random functionals of Ferguson-Dirichlet priors. The Canadian Journal of Statistics, 26(2): 283-297.

See Also

DPMrandom, DPMglmm, DPMlmm, DPlmm , DPglmm, DPolmm, PTlmm , PTglmm, PTolmm

Examples

```
## Not run:

    # Schizophrenia Data
      data(psychiatric)
      attach(psychiatric)

    # MCMC parameters

      nburn<-5000
      nsave<-10000
      nskip<-10
      ndisplay<-100
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay)

    # Initial state
      state <- NULL
    # Prior information

      prior<-list(alpha=1,
                  tau1=0.01,tau2=0.01,
                  nu0=4.01,
                  tinv=diag(10,1),
                  nub=4.01,
                  tbinv=diag(10,1),
                  mb=rep(0,1),
                  Sb=diag(1000,1),
                  beta0=rep(0,3),
                  Sbeta0=diag(1000,3))

    # Fitting the model


      fit1 <- DPMolmm(fixed=imps79o~sweek+tx+sweek*tx,random=~1|id,
                      prior=prior,mcmc=mcmc,state=state,status=TRUE)
      fit1

    # Summary with HPD and Credibility intervals
      summary(fit1)
      summary(fit1,hpd=FALSE)


    # Plot model parameters
      plot(fit1)


    # Plot an specific model parameter
      plot(fit1,ask=FALSE,nfigr=1,nfigc=2,param="sigma-(Intercept)")
      plot(fit1,ask=FALSE,nfigr=1,nfigc=2,param="ncluster")

    # Extract random effects

      DPMrandom(fit1)
      DPMrandom(fit1,centered=TRUE)


    # Extract predictive information of random effects

      pred <- DPMrandom(fit1,predictive=TRUE)
      plot(pred)


## End(Not run)
```

DPMrandom *Extracts Random Effects*

Description

Extracts random effects from DPpackage objects: DPMlmm, DPMolmm, and DPMglmm.

Usage

```
DPMrandom(object,centered=FALSE,predictive=FALSE,
          ngrid=1000,gridl=NULL)
```

Arguments

`object`	DPM fitted model object from which random effects estimates can be extracted.
`centered`	logical variable indicating whether the random effects should be extracted centered, `bi`, or uncentered `thetai`.
`predictive`	logical variable indicating whether actual or predictive information of the random effects should be extracted.
`ngrid`	number of grid points where the density estimate is evaluated. This is only used if dimension of the random effects is lower or equal than 2. The default value is 1000.
`gridl`	The limits of the interval or rectangle covered by the grid as c(xl,xu) or c(xl, xu, yl, yu), respectively. If not specified the grid is defined automatically. This is only used if dimension of the random effects is lower or equal than 2 and if predictive=TRUE.

Author(s)

Alejandro Jara <`<atjara@uc.cl>`>

Examples

```
## Not run:

    # School Girls Data Example

      data(schoolgirls)
      attach(schoolgirls)

    # Prior information

      prior<-list(alpha=1,
                  tau1=0.01,tau2=0.01,
                  nu0=4.01,
                  tinv=diag(10,2),
                  nub=4.01,
                  tbinv=diag(10,2),
                  mb=rep(0,2),
                  Sb=diag(1000,2))

    # Initial state
      state <- NULL
    # MCMC parameters

      nburn<-5000
      nsave<-10000
      nskip<-20
      ndisplay<-1000
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,
                   ndisplay=ndisplay)

    # Fitting the model

      fit1 <- DPMlmm(fixed=height~1,random=~age|child,
                     prior=prior,mcmc=mcmc,
                     state=state,status=TRUE)
      fit1

    # Extract random effects

      DPMrandom(fit1)
      DPMrandom(fit1,centered=TRUE)

      plot(DPMrandom(fit1))
      plot(DPMrandom(fit1,centered=TRUE))

    # Extract predictive information of random effects

      DPMrandom(fit1,predictive=TRUE)
      plot(DPMrandom(fit1,predictive=TRUE,gridl=c(75,89,3.8,7.5)))


## End(Not run)
```

DPMrasch *Bayesian analysis for a semiparametric Rasch model*

Description

This function generates a posterior density sample for a semiparametric Rasch model, using a DPM of normals prior for the distribution of the random effects.

Usage

```
DPMrasch(y,prior,mcmc,offset=NULL,state,status,
         grid=seq(-10,10,length=1000),data=sys.frame(sys.parent()),
         compute.band=FALSE)
```

Arguments

`y`	a matrix giving the data for which the Rasch Model is to be fitted.
`prior`	a list giving the prior information. The list includes the following parameter: `N` giving the truncation of the Dirichlet process prior, `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing), `m0` and `s0` giving the hyperparameters of the normal prior distribution for the mean, `mub`, of the normal baseline distribution, `mub` giving the mean of the baseline distribution (it must be specified if `s0` is missing), `taub1` and `taub2` giving the hyperparameters of the inverted gamma prior distribution for the variance, `sigmab`, of the baseline distribution, `sigmab` giving the variance of the baseline distribution (is must be specified if `taub1` is missing), `tauk1` giving the hyperparameter for the prior distribution of variance of the normal kernel, and `taus1` and `taus2` giving th hyperparameters of the gamma distribution for `tauk2`, `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the difficulty parameters.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`offset`	this can be used to specify an a priori known component to be included in the linear predictor during the fitting.
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`grid`	grid points where the density estimate is evaluated. The default is seq(-10,10,length=1000).
`data`	data frame.
`compute.band`	logical variable indicating whether the confidence band for the density and CDF must be computed.

Details

This generic function fits a semiparametric Rasch model as in San Martin et al. (2011), where

$$\eta_{ij} = \theta_i - \beta_j, i = 1, \ldots, n, j = 1, \ldots, k$$

$$\theta_i | G \sim \int N(\mu, \sigma) G(d\mu, \sigma)$$

$$\beta | \beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

$$G | \alpha, G_0 \sim DP(\alpha G_0)$$

where $G_0 = N(\mu | \mu_b, \sigma_b) IG(\sigma | \tau_{k1}, \tau_{k_2})$. To complete the model specification, independent hyperpriors are assumed,

$$\alpha | a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\mu_b | m0, s0 \sim N(m0, s0)$$

$$\sigma_b^{-2} | \tau_{b1}, \tau_{b2} \sim Gamma(\tau_{b1}/2, \tau_{b2}/2)$$

$$\tau_{k2} | \tau_{s1}, \tau_{s2} \sim Gamma(\tau_{s1}/2, \tau_{s2}/2)$$

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

The computational implementation of the model is based on the finite approximation for DP proposed by Ishwaran and James (2002). The full conditional distributions for the difficulty parameters and in the resampling step of random effects are generated through the Metropolis-Hastings algorithm with a IWLS proposal (see, West, 1985 and Gamerman, 1997).

Value

An object of class `DPMrasch` representing the Rasch model fit. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `beta`, `mub`, `sigmab`, `sigmak2`, the precision parameter `alpha`, and the number of clusters.

The function `DPrandom` can be used to extract the posterior mean of the random effects.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`alpha`	giving the value of the precision parameter.
`b`	a vector of dimension nsubjects giving the value of the random effects for each subject.
`beta`	giving the value of the difficulty parameters.
`ncluster`	an integer giving the number of clusters.
`ss`	an interger vector defining to which of the `ncluster` clusters each subject belongs.
`muclus`	a vector of dimension N giving the value of the normal means.
`sigmaclus`	a vector of dimension N giving the value of the normal variances.
`mub`	giving the mean of the normal baseline distributions.
`sigmab`	giving the variance of the normal baseline distributions.
`tauk2`	giving the parameter of the inverse-gamma prior for the normal kernel variances.
`wdp`	giving the vector of DP weights.
`vdp`	giving the vector of stick-breaking beta random variables used to create the DP weights.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

References

Gamerman, D. (1997) Sampling from the posterior distribution in generalized linear mixed models. Statistics and Computing, 7: 57-68.

Ishwaran, H. and James, L.F. (2002) Approximate Dirichlet process computing finite normal mixtures: smoothing and prior information. Journal of Computational and Graphical Statistics, 11: 508-532.

San Martin, E., Jara, A., Rolin, J.-M., and Mouchart, M. (2011) On the Bayesian nonparametric generalization of IRT-type models. Psychometrika (To appear).

West, M. (1985) Generalized linear models: outlier accomodation, scale parameter and prior distributions. In Bayesian Statistics 2 (eds Bernardo et al.), 531-558, Amsterdam: North Holland.

See Also

DPrandom, DPrasch, FPTrasch

Examples

```
## Not run:
    ####################################
    # A simulated Data Set
    ####################################
      nsubject <- 250
      nitem <- 40

      y <- matrix(0,nrow=nsubject,ncol=nitem)
      dimnames(y)<-list(paste("id",seq(1:nsubject)),
                        paste("item",seq(1,nitem)))

      ind <- rbinom(nsubject,1,0.5)
      theta <- ind*rnorm(nsubject,-1,sqrt(0.25))+
               (1-ind)*rnorm(nsubject,2,sqrt(0.065))
      beta <- c(0,seq(-3,3,length=nitem-1))

      true.density <- function(grid)
      {
            0.5*dnorm(grid,-1,sqrt(0.25))+0.5*dnorm(grid,2,sqrt(0.065))
      }

      true.cdf <- function(grid)
      {
            0.5*pnorm(grid,-1,sqrt(0.25))+0.5*pnorm(grid,2,sqrt(0.065))
      }

      for(i in 1:nsubject)
      {
         for(j in 1:nitem)
         {
            eta <- theta[i]-beta[j]
            prob <- exp(eta)/(1+exp(eta))
            y[i,j] <- rbinom(1,1,prob)
         }
      }

    # Prior information

      beta0 <- rep(0,nitem-1)
      Sbeta0 <- diag(100,nitem-1)

      prior <- list(N=50,
                          alpha=1,
                          taub1=6.01,
                          taub2=2.01,
                          taus1=6.01,
                          taus2=2.01,
                          tauk1=6.01,
                          m0=0,
                          s0=100,
                          beta0=beta0,
                          Sbeta0=Sbeta0)

    # Initial state
      state <- NULL

    # MCMC parameters

      nburn <- 4000
      nsave <- 4000
      nskip <- 0
      ndisplay <- 100
      mcmc <- list(nburn=nburn,
                            nsave=nsave,
                            nskip=nskip,
                            ndisplay=ndisplay)

    # Fit the model
      fit1 <- DPMrasch(y=y,prior=prior,mcmc=mcmc,
                       state=state,status=TRUE,grid=seq(-3,4,0.01))

      plot(fit1$grid,fit1$dens.m,type="l",lty=1,col="red",
           xlim=c(-3,4),ylim=c(0,0.8))
      lines(fit1$grid,true.density(fit1$grid),
            lty=2,col="blue")

      plot(fit1$grid,fit1$cdf.m,type="l",lty=1,col="red")
      lines(fit1$grid,true.cdf(fit1$grid),lty=2,col="blue")

    # Summary with HPD and Credibility intervals
      summary(fit1)
      summary(fit1,hpd=FALSE)

    # Plot model parameters
    # (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE)
```

```
      plot(fit1,ask=FALSE,nfigr=2,nfigc=2)

    # Extract random effects

      DPrandom(fit1)
      plot(DPrandom(fit1))
      DPcaterpillar(DPrandom(fit1))
## End(Not run)
```

DPMraschpoisson *Bayesian analysis for a semiparametric Rasch Poisson model*

Description

This function generates a posterior density sample for a semiparametric Rasch Poisson model, using a DPM of normals prior for the distribution of the random effects.

Usage

```
DPMraschpoisson(y,prior,mcmc,offset=NULL,state,status,
         grid=seq(-10,10,length=1000),data=sys.frame(sys.parent()),
         compute.band=FALSE)
```

Arguments

y	a matrix giving the data for which the Rasch Poisson Model is to be fitted.
prior	a list giving the prior information. The list includes the following parameter: N giving the truncation of the Dirichlet process prior, a0 and b0 giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, alpha giving the value of the precision parameter (it must be specified if a0 is missing), m0 and s0 giving the hyperparameters of the normal prior distribution for the mean, mub, of the normal baseline distribution, mub giving the mean of the baseline distribution (it must be specified if s0 is missing), taub1 and taub2 giving the hyperparameters of the inverted gamma prior distribution for the variance, sigmab, of the baseline distribution, sigmab giving the variance of the baseline distribution (is must be specified if taub1 is missing), tauk1 giving the hyperparameter for the prior distribution of variance of the normal kernel, and taus1 and taus2 giving th hyperparameters of the gamma distribution for tauk2, beta0 and Sbeta0 giving the hyperparameters of the normal prior distribution for the difficulty parameters.
mcmc	a list giving the MCMC parameters. The list must include the following integers: nburn giving the number of burn-in scans, nskip giving the thinning interval, nsave giving the total number of scans to be saved, and ndisplay giving the number of saved scans to be displayed on screen (the function reports on the screen when every ndisplay iterations have been carried out).

`offset`	this can be used to specify an a priori known component to be included in the linear predictor during the fitting.
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`grid`	grid points where the density estimate is evaluated. The default is seq(-10,10,length=1000).
`data`	data frame.
`compute.band`	logical variable indicating whether the confidence band for the density and CDF must be computed.

Details

This generic function fits a semiparametric Rasch Poisson model as in San Martin et al. (2011), where

$$\eta_{ij} = \theta_i - \beta_j, i = 1, \ldots, n, j = 1, \ldots, k$$

$$\theta_i | G \sim \int N(\mu, \sigma) G(d\mu, \sigma)$$

$$\beta | \beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

$$G | \alpha, G_0 \sim DP(\alpha G_0)$$

where $G_0 = N(\mu|\mu_b, \sigma_b) IG(\sigma|\tau_{k1}, \tau_{k_2})$. To complete the model specification, independent hyperpriors are assumed,

$$\alpha | a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\mu_b | m0, s0 \sim N(m0, s0)$$

$$\sigma_b^{-2} | \tau_{b1}, \tau_{b2} \sim Gamma(\tau_{b1}/2, \tau_{b2}/2)$$

$$\tau_{k2} | \tau_{s1}, \tau_{s2} \sim Gamma(\tau_{s1}/2, \tau_{s2}/2)$$

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

The computational implementation of the model is based on the finite approximation for DP proposed by Ishwaran and James (2002). The full conditional distributions for the difficulty parameters and in the resampling step of random effects are generated through the Metropolis-Hastings algorithm with a IWLS proposal (see, West, 1985 and Gamerman, 1997).

Value

An object of class `DPMrasch` representing the Rasch model fit. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `beta`, `mub`, `sigmab`, `sigmak2`, the precision parameter `alpha`, and the number of clusters.

The function `DPrandom` can be used to extract the posterior mean of the random effects.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

alpha	giving the value of the precision parameter.
b	a vector of dimension nsubjects giving the value of the random effects for each subject.
beta	giving the value of the difficulty parameters.
ncluster	an integer giving the number of clusters.
ss	an interger vector defining to which of the `ncluster` clusters each subject belongs.
muclus	a vector of dimension N giving the value of the normal means.
sigmaclus	a vector of dimension N giving the value of the normal variances.
mub	giving the mean of the normal baseline distributions.
sigmab	giving the variance of the normal baseline distributions.
tauk2	giving the parameter of the inverse-gamma prior for the normal kernel variances.
wdp	giving the vector of DP weights.
vdp	giving the vector of stick-breaking beta random variables used to create the DP weights.

Author(s)

Alejandro Jara <<atjara@uc.cl>>

References

Gamerman, D. (1997) Sampling from the posterior distribution in generalized linear mixed models. Statistics and Computing, 7: 57-68.

Ishwaran, H. and James, L.F. (2002) Approximate Dirichlet process computing finite normal mixtures: smoothing and prior information. Journal of Computational and Graphical Statistics, 11:508-532.

San Martin, E., Jara, A., Rolin, J.-M., and Mouchart, M. (2011) On the Bayesian nonparametric generalization of IRT-type models. Psychometrika (To appear)

West, M. (1985) Generalized linear models: outlier accomodation, scale parameter and prior distributions. In Bayesian Statistics 2 (eds Bernardo et al.), 531-558, Amsterdam: North Holland.

See Also

DPrandom, DPraschpoisson, FPTraschpoisson

Examples

```
## Not run:
    ####################################
    # A simulated Data Set
    ####################################
      nsubject <- 250
      nitem <- 2

      y <- matrix(0,nrow=nsubject,ncol=nitem)
```

```
  dimnames(y)<-list(paste("id",seq(1:nsubject)),
                                  paste("item",seq(1,nitem)))

  ind <- rbinom(nsubject,1,0.5)
  theta <- ind*rnorm(nsubject,-1,0.5)+(1-ind)*rnorm(nsubject,2,0.25)
  beta <- c(0,seq(-3,3,length=nitem-1))

  true.density <- function(grid)
  {
        0.5*dnorm(grid,-1,0.5)+0.5*dnorm(grid,2,0.25)
  }

  true.cdf <- function(grid)
  {
        0.5*pnorm(grid,-1,0.5)+0.5*pnorm(grid,2,0.25)
  }

  for(i in 1:nsubject)
  {
     for(j in 1:nitem)
     {
        eta <- theta[i]-beta[j]
        rate <- exp(eta)
        y[i,j] <- rpois(1,rate)
     }
  }

# Prior information

  beta0 <- rep(0,nitem-1)
  Sbeta0 <- diag(100,nitem-1)

  prior <- list(N=50,
                    a0=2,
                    b0=0.1,
                    taub1=6.01,
                    taub2=2.01,
                    taus1=6.01,
                    taus2=2.01,
                    tauk1=6.01,
                    m0=0,
                    s0=100,
                    beta0=beta0,
                    Sbeta0=Sbeta0)

# Initial state
  state <- NULL

# MCMC parameters

  nburn <- 5000
  nsave <- 5000
  nskip <- 0
```

```
      ndisplay <- 100
      mcmc <- list(nburn=nburn,
                         nsave=nsave,
                         nskip=nskip,
                         ndisplay=ndisplay)

    # Fit the model
      fit1 <- DPMraschpoisson(y=y,prior=prior,mcmc=mcmc,
                          state=state,status=TRUE,grid=seq(-3,4,0.01))

      plot(fit1$grid,fit1$dens.m,type="l",lty=1,col="red",
           xlim=c(-3,4),ylim=c(0,0.8))
      lines(fit1$grid,true.density(fit1$grid),lty=2,col="blue")

      plot(fit1$grid,fit1$cdf.m,type="l",lty=1,col="red")
      lines(fit1$grid,true.cdf(fit1$grid),lty=2,col="blue")

    # Summary with HPD and Credibility intervals
      summary(fit1)
      summary(fit1,hpd=FALSE)

    # Plot model parameters
    # (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE)
      plot(fit1,ask=FALSE,nfigr=2,nfigc=2)

    # Extract random effects

      DPrandom(fit1)
      plot(DPrandom(fit1))
      DPcaterpillar(DPrandom(fit1))

## End(Not run)
```

DPmultmeta — *Bayesian analysis for a semiparametric random effects multivariate meta-analysis model using a MDP*

Description

This function generates a posterior density sample for a semiparametric random effects multivariate meta-analysis model using a Dirichlet process or a Mixture of Dirichlet process prior for the distribution of the random effects. Support provided by the NIH/NCI R01CA75981 grant.

Usage

```
DPmultmeta(y,asymvar,prior,mcmc,state,status,
            data=sys.frame(sys.parent()))
```

Arguments

`y`	a vector or matrix giving the data or effects from which the density estimate is to be computed.
`asymvar`	a vactor or matrix giving the asymptotic covariance matrix for each effect. The dimension of this matrix is the number of records/studies times the the half-stored elements of the study-specific covariance matrix.
`prior`	a list giving the prior information. The list includes the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing, see details below), `m2` and `s2` giving the mean and the covariance of the normal prior for the mean, `m1`, of the mean of the normal baseline distribution, respectively, `m1` giving the mean of the baseline distribution (it must be specified if `m2` is missing), `nu` and `psiinv` giving the hyperparameters of the inverted Wishart distribution on the covariance matrix `s1` of the normal baseline distribution, and `s1` giving the covariance matrix of the baseline distribution (it must be specified if `nu` is missing).
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.

Details

This generic function fits a semiparametric random effects multivariate meta-analysis model:

$$y_i \sim N(\mu_i, \Sigma_i), i = 1, \ldots, n$$

$$\theta_i | G \sim G$$

$$G | \alpha, G_0 \sim DP(\alpha G_0)$$

where, $G_0 = N(\theta | m_1, s_1)$. To complete the model specification, independent hyperpriors are assumed,

$$\alpha | a_0, b_0 \sim Gamma(a_0, b_0)$$

$$m_1 | m_2, s_2 \sim N(mu_2, s_2)$$

$$s_1 | \nu, \psi \sim IW(\nu, \psi)$$

Note that the inverted-Wishart prior is parametrized such that if $A \sim IW_q(\nu, \psi)$ then $E(A) = \psi^{-1}/(\nu - q - 1)$.

To let part of the baseline distribution fixed at a particular value, set the corresponding hyperparameters of the prior distributions to NULL in the hyperprior specification of the model.

The computational implementation of the model is based on the marginalization of the `DP` and on the use of MCMC methods for conjugate priors for a collapsed state of MacEachern (1998).

Value

An object of class `DPmultmeta` representing the random effects model fit. Generic functions such as `print`, `plot`, and `summary`, have methods to show the results of the fit. The results include `m1`, `s1`, `alpha`, and the number of clusters.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`ncluster`	an integer giving the number of clusters.
`alpha`	giving the value of the precision parameter
`muclus`	a matrix of dimension (nobservations+1)*(nvariables) giving the means of the clusters (only the first `ncluster` are considered to start the chain).
`ss`	an interger vector defining to which of the `ncluster` clusters each subject belongs.
`m1`	giving the mean of the normal baseline distributions.
`s1`	giving the covariance matrix of the normal baseline distributions.

Author(s)

Alejandro Jara <`<atjara@uc.cl>`>

Peter Mueller <`<pmueller@mdanderson.org>`>

References

MacEachern, S.N. (1998) Computational Methods for Mixture of Dirichlet Process Models, in Practical Nonparametric and Semiparametric Bayesian Statistics, eds: D. Dey, P. Muller, D. Sinha, New York: Springer-Verlag, pp. 23-44.

See Also

`DPmeta`

Examples

```
## Not run:
    #########################################################################
    # Simulated Data:
    #         mu_i ~ 0.5 N(mub1,Sigmab1) + 0.5 N(mub2,Sigmab2)
    #         y_i ~ N(mu_i,Sigma_i)
    #         Sigma_1=...=Sigma_n=Sigma assumed to be known
    #########################################################################
```

```
  nvar <- 2
  nrec <- 100
  Sigma <- matrix(c(0.25,0.15,0.15,0.25),nrow=nvar,ncol=nvar)
  mub1 <- rep(-1.5,nvar)
  mub2 <- rep( 0.5,nvar)
  Sigmab1 <- matrix(c(0.25,-0.175,-0.175,0.25),nrow=nvar,ncol=nvar)
  Sigmab2 <- matrix(c(0.25, 0.0875, 0.0875,0.25),nrow=nvar,ncol=nvar)

  ind <- rbinom(nrec,1,0.5)
  z1 <- mub1+matrix(rnorm(nvar*nrec),nrow=nrec,ncol=nvar)
  z2 <- mub2+matrix(rnorm(nvar*nrec),nrow=nrec,ncol=nvar)
  mu <- ind*z1+(1-ind)*z2

  y <- NULL
  for(i in 1:nrec)
  {
      z <- mu[i,]+matrix(rnorm(nvar),nrow=1,ncol=nvar)
      y <- rbind(y,z)

  }
  colnames(y) <- c("y1","y2")

#######################################################################
# Asymptotic variance
#######################################################################
  z <- NULL
  for(i in 1:nvar)
  {
      for(j in i:nvar)
      {
            z <- c(z,Sigma[i,j])
      }
  }
  asymvar <- matrix(z,nrow=nrec,ncol=nvar*(nvar+1)/2,byrow=TRUE)


# Prior information

  s2 <-diag(100,nvar)
  m2 <-rep(0,nvar)
  nu <- 4
  psiinv <- diag(1,nvar)

  prior<-list(a0=1,
              b0=1/5,
              nu=nu,
              m2=m2,
              s2=s2,
              psiinv=psiinv)

# Initial state
  state <- NULL
```

```
    # MCMC parameters

      nburn <- 500
      nsave <- 1000
      nskip <- 0
      ndisplay <- 100
      mcmc <- list(nburn=nburn,
                   nsave=nsave,
                   nskip=nskip,
                   ndisplay=ndisplay)

    # Fitting the model
      fit1 <- DPmultmeta(y=y,asymvar=asymvar,prior=prior,
                         mcmc=mcmc,state=state,status=TRUE)


## End(Not run)
```

DPolmm *Bayesian analysis for a semiparametric ordinal linear mixed model using a MDP*

Description

This function generates a posterior density sample for a semiparametric ordinal linear mixed model using a Dirichlet Process or a Mixture of Dirichlet process prior for the distribution of the random effects.

Usage

```
DPolmm(fixed,random,prior,mcmc,state,status,
       data=sys.frame(sys.parent()),
       na.action=na.fail)
```

Arguments

`fixed`	a two-sided linear formula object describing the fixed-effects part of the model, with the response on the left of a ~ operator and the terms, separated by + operators, on the right.
`random`	a one-sided formula of the form `~z1+...+zn	g`, with `z1+...+zn` specifying the model for the random effects and `g` the grouping variable. The random effects formula will be repeated for all levels of grouping.
`prior`	a list giving the prior information. The list include the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` and `b0` are missing, see details below), `nu0` and `Tinv` giving the hyperparameters of the inverted Wishart prior distribution

	for the scale matrix of the normal baseline distribution, `sigma` giving the value of the covariance matrix of the centering distribution (it must be specified if `nu0` and `tinv` are missing), `mub` and `Sb` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `mu` giving the value of the mean of the centering distribution (it must be specified if `mub` and `Sb` are missing), `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the fixed effects (must be specified only if fixed effects are considered in the model).
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `DPolmm` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits an ordinal linear mixed-effects model with a probit link (see, e.g., Molenberghs and Verbeke, 2005):

$$Y_{ij} = k, \text{ if } \gamma_{k-1} \le W_{ij} < \gamma_k, k = 1, \ldots, K$$

$$W_{ij} \mid \beta_F, \beta_R, b_i \sim N(X_{ij}\beta_F + Z_{ij}\beta_R + Z_{ij}b_i, 1), i = 1, \ldots, N, j = 1, \ldots, n_i$$

$$\theta_i | G \sim G$$

$$G | \alpha, G_0 \sim DP(\alpha G_0)$$

where, $\theta_i = \beta_R + b_i$, $\beta = \beta_F$, and $G_0 = N(\theta | \mu, \Sigma)$. To complete the model specification, independent hyperpriors are assumed,

$$\alpha | a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\beta | \beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

$$\mu | \mu_b, S_b \sim N(\mu_b, S_b)$$

$$\Sigma | \nu_0, T \sim IW(\nu_0, T)$$

A uniform prior is used for the cutoff points. Note that the inverted-Wishart prior is parametrized such that $E(\Sigma) = T^{-1}/(\nu_0 - q - 1)$.

The precision or total mass parameter, α, of the DP prior can be considered as random, having a gamma distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

The computational implementation of the model is based on the marginalization of the DP and on the use of MCMC methods for conjugate priors (Escobar, 1994; Escobar and West, 1998). The β_R parameters are sampled using the ϵ-DP approximation proposed by Muliere and Tardella (1998), with ϵ=0.01.

Value

An object of class `DPolmm` representing the linear mixed-effects model fit. Generic functions such as `print`, `plot`, `summary`, and `anova` have methods to show the results of the fit. The results include `betaR`, `betaF`, `mu`, the elements of `Sigma`, the cutoff points, `alpha`, and the number of clusters.

The function `DPrandom` can be used to extract the posterior mean of the random effects.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`ncluster`	an integer giving the number of clusters.
`alpha`	giving the value of the precision parameter
`b`	a matrix of dimension (nsubjects)*(nrandom effects) giving the value of the random effects for each subject.
`bclus`	a matrix of dimension (nsubjects)*(nrandom effects) giving the value of the random effects for each clusters (only the first `ncluster` are considered to start the chain).
`cutoff`	a real vector defining the cutoff points. Note that the first cutoff must be fixed to 0 in this function.
`ss`	an interger vector defining to which of the `ncluster` clusters each subject belongs.
`beta`	giving the value of the fixed effects.
`mu`	giving the mean of the normal baseline distributions.
`sigma`	giving the variance matrix of the normal baseline distributions.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

References

Escobar, M.D. (1994) Estimating Normal Means with a Dirichlet Process Prior, Journal of the American Statistical Association, 89: 268-277.

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Escobar, M.D. and West, M. (1998) Computing Bayesian Nonparametric Hierarchical Models, in Practical Nonparametric and Semiparametric Bayesian Statistics, eds: D. Dey, P. Muller, D. Sinha, New York: Springer-Verlag, pp. 1-22.

Molenberghs, G. and Verbeke, G. (2005). Models for discrete longitudinal data, New York: Springer-Verlag.

Muliere, P. and Tardella, L. (1998) Approximating distributions of random functionals of Ferguson-Dirichlet priors. The Canadian Journal of Statistics, 26(2): 283-297.

See Also

DPrandom, DPlmm , DPglmm, DPMglmm, DPMlmm, DPMolmm, PTlmm , PTglmm, PTolmm

Examples

```
## Not run:

    # Schizophrenia Data
      data(psychiatric)
      attach(psychiatric)

    # MCMC parameters

      nburn<-5000
      nsave<-10000
      nskip<-10
      ndisplay<-100
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay)

    # Initial state
      state <- NULL


    # Prior information

      tinv<-diag(10,1)
      prior<-list(alpha=1,nu0=4.01,tinv=tinv,mub=rep(0,1),Sb=diag(100,1),
                  beta0=rep(0,3),Sbeta0=diag(1000,3))


    # Fitting the model


      fit1<-DPolmm(fixed=imps79o~sweek+tx+sweek*tx,random=~1|id,
                   prior=prior,mcmc=mcmc,state=state,status=TRUE)
      fit1

    # Summary with HPD and Credibility intervals
      summary(fit1)
      summary(fit1,hpd=FALSE)

    # Plot model parameters
```

```
      plot(fit1)

    # Plot an specific model parameter
      plot(fit1,ask=FALSE,nfigr=1,nfigc=2,param="sigma-(Intercept)")
      plot(fit1,ask=FALSE,nfigr=1,nfigc=2,param="ncluster")

    # Extract random effects
      DPrandom(fit1)
      DPrandom(fit1,centered=TRUE)

    # Extract predictive information of random effects
      DPrandom(fit1,predictive=TRUE)
      DPrandom(fit1,centered=TRUE,predictive=TRUE)

      plot(DPrandom(fit1,predictive=TRUE))
      plot(DPrandom(fit1,centered=TRUE,predictive=TRUE))

## End(Not run)
```

DPpsBF *Computes Pseudo Bayes Factors from DPpackage output*

Description

This function computes Pseudo Bayes Factors from DPpackage output.

Usage

```
DPpsBF(...)
```

Arguments

`...`	DPpackage output objects. These have to be of the same class.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

Examples

```
## Not run:
    # Respiratory Data Example

      data(indon)
      attach(indon)

      baseage2 <- baseage**2
      follow <- age-baseage
      follow2 <- follow**2
    # Prior information

      beta0 <- rep(0,9)
      Sbeta0 <- diag(1000,9)
      tinv <- diag(1,1)
      prior <- list(a0=2,b0=0.1,nu0=4,tinv=tinv,
                    mub=rep(0,1),Sb=diag(1000,1),
                    beta0=beta0,Sbeta0=Sbeta0)

    # Initial state
      state <- NULL

    # MCMC parameters

      nburn <- 5
      nsave <- 100
      nskip <- 5
      ndisplay <- 100
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay)

    # Fit the Probit model
      fit1 <- DPglmm(fixed=infect~gender+height+cosv+sinv+xero+baseage+
                     baseage2+follow+follow2,random=~1|id,
                     family=binomial(probit),
                     prior=prior,mcmc=mcmc,state=state,status=TRUE)

    # Fit the Logit model
      fit2 <- DPglmm(fixed=infect~gender+height+cosv+sinv+xero+baseage+
                     baseage2+follow+follow2,random=~1|id,
                     family=binomial(logit),
                     prior=prior,mcmc=mcmc,state=state,status=TRUE)

    # Model comparison
      DPpsBF(fit1,fit2)


## End(Not run)
```

DPrandom *Extracts Random Effects*

Description

This generic function extracts Random Effects' information from DPpackage model objects.

Usage

```
DPrandom(object,centered=FALSE,predictive=FALSE)
```

Arguments

`object`	DP fitted model object from which random effects estimates can be extracted.
`centered`	logical variable indicating whether the random effects should be extracted centered, `bi`, or uncentered `thetai`.
`predictive`	logical variable indicating whether actual or predictive information of the random effects should be extracted.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

Examples

```
## Not run:
    # School Girls Data Example

      data(schoolgirls)
      attach(schoolgirls)

    # Prior information
    # Prior information

      tinv<-diag(10,2)
      prior<-list(alpha=1,nu0=4.01,tau1=0.001,tau2=0.001,
      tinv=tinv,mub=rep(0,2),Sb=diag(1000,2))

    # Initial state
      state <- NULL

    # MCMC parameters

      nburn<-5000
      nsave<-25000
      nskip<-20
      ndisplay<-1000
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay)

    # Fit the model

      fit1<-DPlmm(fixed=height~1,random=~age|child,prior=prior,mcmc=mcmc,
                  state=state,status=TRUE)
      fit1


    # Extract random effects

      DPrandom(fit1)
      DPrandom(fit1,centered=TRUE)

      plot(DPrandom(fit1))
      plot(DPrandom(fit1,centered=TRUE))
    # Extract predictive information of random effects

      DPrandom(fit1,predictive=TRUE)
      DPrandom(fit1,centered=TRUE,predictive=TRUE)

      plot(DPrandom(fit1,predictive=TRUE))
      plot(DPrandom(fit1,centered=TRUE,predictive=TRUE))

## End(Not run)
```

DPrasch — *Bayesian analysis for a semiparametric Rasch model*

Description

This function generates a posterior density sample for a semiparametric Rasch model, using a DP or a MDP prior for the distribution of the random effects.

Usage

```
DPrasch(y,prior,mcmc,offset,state,status,
        grid=seq(-10,10,length=1000),data=sys.frame(sys.parent()),
        compute.band=FALSE)
```

Arguments

`y`	a matrix giving the data for which the Rasch Model is to be fitted.
`prior`	a list giving the prior information. The list includes the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing), `mub` and `Sb` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `mu` giving the mean of the normal baseline distribution (is must be specified if `mub` and `Sb` are missing), `tau1` and `tau2` giving the hyperparameters for the prior distribution of variance of the normal baseline distribution, `sigma2` giving the variance of the normal baseline distribution (is must be specified if `tau1` and `tau2` are missing), and `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the difficulty parameters.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).

`offset`	this can be used to specify an a priori known component to be included in the linear predictor during the fitting.
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`grid`	grid points where the density estimate is evaluated. The default is seq(-10,10,length=1000).
`data`	data frame.
`compute.band`	logical variable indicating whether the confidence band for the CDF must be computed.

Details

This generic function fits a semiparametric Rasch model as in San Martin et al. (2011), where the linear predictor is modeled as follows:

$$\eta_{ij} = \theta_i - \beta_j, i = 1, \ldots, n, j = 1, \ldots, k$$

$$\theta_i | G \sim G$$

$$G | \alpha, G_0 \sim DP(\alpha G_0)$$

$$\beta | \beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

where, $G_0 = N(\theta|\mu, \sigma^2)$. To complete the model specification, independent hyperpriors are assumed,

$$\alpha | a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\mu | \mu_b, S_b \sim N(\mu_b, S_b)$$

$$\sigma^{-2} | \tau_1, \tau_2 \sim Gamma(\tau_1/2, \tau_2/2)$$

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

Each of the parameters of the baseline distribution, μ and σ^2 can be considered as random or fixed at some particular value. In the first case, a Mixture of Dirichlet Process is considered as a prior for the distribution of the random effects. To let σ^2 to be fixed at a particular value, set τ_1 to NULL in the prior specification. To let μ to be fixed at a particular value, set μ_b to NULL in the prior specification.

The computational implementation of the model is based on the marginalization of the `DP` and on the use of MCMC methods for nonconjugate priors (see, MacEachern and Muller, 1998; Neal, 2000). Specifically, the algorithm 8 with `m=1` of Neal (2000), is considered in the `DPraschpoisson` function. In this case, the full conditional distributions for the difficulty parameters and in the resampling step of random effects are generated through the Metropolis-Hastings algorithm with a IWLS proposal (see, West, 1985 and Gamerman, 1997).

The functionals parameters are sampled using the ϵ-DP approximation proposed by Muliere and Tardella (1998), with ϵ=0.01.

Value

An object of class `DPrasch` representing the Rasch model fit. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `beta`, `mu`, `sigma2`, the precision parameter `alpha`, and the number of clusters.

The function `DPrandom` can be used to extract the posterior mean of the random effects.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`ncluster`	an integer giving the number of clusters.
`alpha`	giving the value of the precision parameter.
`b`	a vector of dimension nsubjects giving the value of the random effects for each subject.
`bclus`	a vector of dimension nsubjects giving the value of the random effects for each clusters (only the first `ncluster` are considered to start the chain).
`ss`	an interger vector defining to which of the `ncluster` clusters each subject belongs.
`beta`	giving the value of the difficulty parameters.
`mu`	giving the mean of the normal baseline distributions.
`sigma2`	giving the variance of the normal baseline distributions.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

References

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Gamerman, D. (1997) Sampling from the posterior distribution in generalized linear mixed models. Statistics and Computing, 7: 57-68.

MacEachern, S. N. and Muller, P. (1998) Estimating mixture of Dirichlet Process Models. Journal of Computational and Graphical Statistics, 7 (2): 223-338.

Muliere, P. and Tardella, L. (1998) Approximating distributions of random functionals of Ferguson-Dirichlet priors. The Canadian Journal of Statistics, 26(2): 283-297.

Neal, R. M. (2000) Markov Chain sampling methods for Dirichlet process mixture models. Journal of Computational and Graphical Statistics, 9:249-265.

San Martin, E., Jara, A., Rolin, J.-M., and Mouchart, M. (2011) On the Bayesian nonparametric generalization of IRT-type models. Psychometrika (To appear)

West, M. (1985) Generalized linear models: outlier accomodation, scale parameter and prior distributions. In Bayesian Statistics 2 (eds Bernardo et al.), 531-558, Amsterdam: North Holland.

See Also

DPrandom, FPTrasch

Examples

```
## Not run:
    ####################################
    # A simulated Data Set
    ####################################
      nsubject <- 200
      nitem <- 40

      y <- matrix(0,nrow=nsubject,ncol=nitem)
      dimnames(y) <- list(paste("id",seq(1:nsubject)),
                          paste("item",seq(1,nitem)))


      ind <- rbinom(nsubject,1,0.5)
      theta <- ind*rnorm(nsubject,1,0.25)+(1-ind)*rnorm(nsubject,3,0.25)
      beta <- c(0,seq(-1,3,length=nitem-1))
      true.cdf <- function(grid)
      {
         0.5*pnorm(grid,1,0.25)+0.5*pnorm(grid,3,0.25)
      }
      for(i in 1:nsubject)
      {
         for(j in 1:nitem)
         {
            eta<-theta[i]-beta[j]
            mean<-exp(eta)/(1+exp(eta))
            y[i,j]<-rbinom(1,1,mean)
         }
      }

    # Prior information

      beta0 <- rep(0,nitem-1)
      Sbeta0 <- diag(1000,nitem-1)

      prior <- list(alpha=1,
                    tau1=6.02,
                    tau2=2.02,
                    mub=0,
                    Sb=100,
                    beta0=beta0,
                    Sbeta0=Sbeta0)

    # Initial state
      state <- NULL

    # MCMC parameters
      nburn <- 5000
      nsave <- 5000
      nskip <- 0
      ndisplay<- 1000
      mcmc <- list(nburn=nburn,
                   nsave=nsave,
                   nskip=nskip,
                   ndisplay=ndisplay)

    # Fit the model
      fit1 <- DPrasch(y=y,prior=prior,mcmc=mcmc,
                      state=state,status=TRUE,grid=seq(-1,5,0.01),
                      compute.band=TRUE)

    # CDF estimate and truth
      plot(fit1$grid,true.cdf(fit1$grid),type="l",lwd=2,col="red",
           xlab=expression(theta),ylab="CDF")
      lines(fit1$grid,fit1$cdf,lwd=2,col="blue")
      lines(fit1$grid,fit1$cdf.l,lwd=2,col="blue",lty=2)
      lines(fit1$grid,fit1$cdf.u,lwd=2,col="blue",lty=2)

    # Summary with HPD and Credibility intervals
      summary(fit1)
      summary(fit1,hpd=FALSE)

    # Plot model parameters
    # (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE)
      plot(fit1,ask=FALSE,nfigr=2,nfigc=2)

    # Extract random effects

      DPrandom(fit1)
      plot(DPrandom(fit1))
      DPcaterpillar(DPrandom(fit1))

## End(Not run)
```

DPraschpoisson — *Bayesian analysis for a semiparametric Rasch Poisson model*

Description

This function generates a posterior density sample for a semiparametric Rasch Poisson model, using a DP or a MDP prior for the distribution of the random effects.

Usage

```
DPraschpoisson(y,prior,mcmc,offset,state,status,
          grid=seq(-10,10,length=1000),data=sys.frame(sys.parent()),
          compute.band=FALSE)
```

Arguments

`y`	a matrix giving the data for which the Rasch Poisson Model is to be fitted.
`prior`	a list giving the prior information. The list includes the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing), `mub` and `Sb` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `mu` giving the mean of the normal baseline distribution (is must be specified if `mub` and `Sb` are missing), `tau1` and `tau2` giving the hyperparameters for the prior distribution of variance of the normal baseline distribution, `sigma2` giving the variance of the normal baseline distribution (is must be specified if `tau1` and `tau2` are missing), and `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the difficulty parameters.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`offset`	this can be used to specify an a priori known component to be included in the linear predictor during the fitting.
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`grid`	grid points where the density estimate is evaluated. The default is seq(-10,10,length=1000).
`data`	data frame.
`compute.band`	logical variable indicating whether the confidence band for the CDF must be computed.

Details

This generic function fits a semiparametric Rasch Poisson model as in San Martin et al. (2011), where the linear predictor is modeled as follows:

$$\eta_{ij} = \theta_i - \beta_j, i = 1, \ldots, n, j = 1, \ldots, k$$

$$\theta_i | G \sim G$$

$$G | \alpha, G_0 \sim DP(\alpha G_0)$$

$$\beta | \beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

where, $G_0 = N(\theta|\mu, \sigma^2)$. To complete the model specification, independent hyperpriors are assumed,

$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$
$$\mu|\mu_b, S_b \sim N(\mu_b, S_b)$$
$$\sigma^{-2}|\tau_1, \tau_2 \sim Gamma(\tau_1/2, \tau_2/2)$$

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

Each of the parameters of the baseline distribution, μ and σ^2 can be considered as random or fixed at some particular value. In the first case, a Mixture of Dirichlet Process is considered as a prior for the distribution of the random effects. To let σ^2 to be fixed at a particular value, set τ_1 to NULL in the prior specification. To let μ to be fixed at a particular value, set μ_b to NULL in the prior specification.

The computational implementation of the model is based on the marginalization of the `DP` and on the use of MCMC methods for nonconjugate priors (see, MacEachern and Muller, 1998; Neal, 2000). Specifically, the algorithm 8 with `m=1` of Neal (2000), is considered in the `DPraschpoisson` function. In this case, the fully conditional distributions for the difficulty parameters and in the resampling step of random effects are generated through the Metropolis-Hastings algorithm with a IWLS proposal (see, West, 1985 and Gamerman, 1997).

The functionals parameters are sampled using the ϵ-DP approximation proposed by Muliere and Tardella (1998), with ϵ=0.01.

Value

An object of class `DPraschpoisson` representing the Rasch Poisson model fit. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `beta`, `mu`, `sigma2`, the precision parameter `alpha`, and the number of clusters.

The function `DPrandom` can be used to extract the posterior mean of the random effects.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`ncluster`	an integer giving the number of clusters.
`alpha`	giving the value of the precision parameter.
`b`	a vector of dimension nsubjects giving the value of the random effects for each subject.
`bclus`	a vector of dimension nsubjects giving the value of the random effects for each clusters (only the first `ncluster` are considered to start the chain).
`ss`	an interger vector defining to which of the `ncluster` clusters each subject belongs.
`beta`	giving the value of the difficulty parameters.
`mu`	giving the mean of the normal baseline distributions.
`sigma2`	giving the variance of the normal baseline distributions.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

References

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Gamerman, D. (1997) Sampling from the posterior distribution in generalized linear mixed models. Statistics and Computing, 7: 57-68.

MacEachern, S. N. and Muller, P. (1998) Estimating mixture of Dirichlet Process Models. Journal of Computational and Graphical Statistics, 7 (2): 223-338.

Muliere, P. and Tardella, L. (1998) Approximating distributions of random functionals of Ferguson-Dirichlet priors. The Canadian Journal of Statistics, 26(2): 283-297.

Neal, R. M. (2000) Markov Chain sampling methods for Dirichlet process mixture models. Journal of Computational and Graphical Statistics, 9:249-265.

San Martin, E., Jara, A., Rolin, J.-M., and Mouchart, M. (2011) On the Bayesian nonparametric generalization of IRT-type models. Psychometrika (To appear)

West, M. (1985) Generalized linear models: outlier accomodation, scale parameter and prior distributions. In Bayesian Statistics 2 (eds Bernardo et al.), 531-558, Amsterdam: North Holland.

See Also

`DPrandom`, `FPTraschpoisson`

Examples

```
## Not run:
    ####################################
    # A simulated Data Set
    ####################################
      nsubject <- 200
      nitem <- 5

      y <- matrix(0,nrow=nsubject,ncol=nitem)

      ind <- rbinom(nsubject,1,0.5)
      theta <- ind*rnorm(nsubject,1,0.25)+(1-ind)*rnorm(nsubject,3,0.25)
      beta <- c(0,seq(-1,1,length=nitem-1))
      true.cdf <- function(grid)
      {
         0.5*pnorm(grid,1,0.25)+0.5*pnorm(grid,3,0.25)
      }

      for(i in 1:nsubject)
      {
         for(j in 1:nitem)
         {
            eta <- theta[i]-beta[j]
            means <- exp(eta)
```

```
        y[i,j] <- rpois(1,means)
     }
  }

# Prior information

  beta0 <- rep(0,nitem-1)
  Sbeta0 <- diag(1000,nitem-1)

  prior <- list(alpha=1,
                tau1=6.02,
                tau2=2.02,
                mub=0,
                Sb=100,
                beta0=beta0,
                Sbeta0=Sbeta0)

# Initial state
  state <- NULL

# MCMC parameters

  nburn <- 5000
  nsave <- 5000
  nskip <- 0
  ndisplay<- 1000
  mcmc <- list(nburn=nburn,
               nsave=nsave,
               nskip=nskip,
               ndisplay=ndisplay)

# Fit the model
  fit1 <- DPraschpoisson(y=y,prior=prior,mcmc=mcmc,
                         state=state,status=TRUE,grid=seq(-1,5,0.01),
                         compute.band=TRUE)

# CDF estimate and true
  plot(fit1$grid,true.cdf(fit1$grid),type="l",lwd=2,col="red",
       xlab=expression(theta),ylab="CDF")
  lines(fit1$grid,fit1$cdf,lwd=2,col="blue")
  lines(fit1$grid,fit1$cdf.l,lwd=2,col="blue",lty=2)
  lines(fit1$grid,fit1$cdf.u,lwd=2,col="blue",lty=2)

# Summary with HPD and Credibility intervals
  summary(fit1)
  summary(fit1,hpd=FALSE)

# Plot model parameters
# (to see the plots gradually set ask=TRUE)
  plot(fit1,ask=FALSE)
  plot(fit1,ask=FALSE,nfigr=2,nfigc=2)

# Extract random effects
```

```
    DPrandom(fit1)
    plot(DPrandom(fit1))
    DPcaterpillar(DPrandom(fit1))


## End(Not run)
```

DProc *Semiparametric Bayesian ROC curve analysis using DPM of normals*

Description

This function performs a ROC curve analysis based on a posterior density sample for Dirichlet process mixture of normals models.

Usage

```
DProc(x,y,fitx=NULL,fity=NULL,ngrid=1000,priorx,priory,
      mcmcx,mcmcy,statex,statey,
      statusx,statusy,data=sys.frame(sys.parent()),
      na.action=na.fail)
```

Arguments

`x`	a vector giving the diagnostic marker measurements for the healthy subjects.
`y`	a vector giving the diagnostic marker measurements for the diseased subjects.
`fitx`	a object containing the results returned by the DPdensity model fitting function for the diagnostic marker measurements in the healthy subjects (Optional).
`fity`	a object containing the results returned by the DPdensity model fitting function for the diagnostic marker measurements in the diseased subjects (Optional).
`ngrid`	number of grid points where the ROC curve is evaluated. The default value is 1000.
`priorx`	a list giving the prior information for the diagnostic marker measurements in the healthy subjects. See the DPdensity function for details. (it must be specified if `fitx` is missing).
`priory`	a list giving the prior information for the diagnostic marker measurements in the diseased subjects. See the DPdensity function for details. (it must be specified if `fity` is missing).
`mcmcx`	a list giving the MCMC parameters for the diagnostic marker measurements in the healthy subjects. See the DPdensity function for details. (it must be specified if `fitx` is missing).
`mcmcy`	a list giving the MCMC parameters for the diagnostic marker measurements in the diseased subjects. See the DPdensity function for details. (it must be specified if `fity` is missing).

`statex`	a list giving the current value of the parameters. See the DPdensity function for details. (it must be specified if `fitx` is missing).
`statey`	a list giving the current value of the parameters. See the DPdensity function for details. (it must be specified if `fity` is missing).
`statusx`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `statex`.
`statusy`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `statex`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `DProc` to print an error message and terminate if there are any incomplete observations.

Details

This generic function performs a ROC curve analysis based on Dirichlet process mixture of normals models for density estimation (Escobar and West, 1995):

$$x_i|\mu_{x_i}, \Sigma_{x_i} \sim N(\mu_{x_i}, \Sigma_{x_i}), i = 1, \ldots, n$$

$$(\mu_{x_i}, \Sigma_{x_i})|G_x \sim G_x$$

$$G_x|\alpha_x, G_{x_0} \sim DP(\alpha_x G_{x_0})$$

$$y_j|\mu_{y_j}, \Sigma_{y_j} \sim N(\mu_{y_j}, \Sigma_{y_j}), j = 1, \ldots, m$$

$$(\mu_{y_j}, \Sigma_{y_j})|G_y \sim G_y$$

$$G_y|\alpha_y, G_{y_0} \sim DP(\alpha_y G_{y_0})$$

where, x and y is the vector containing the diagnostic marker measurements in the healthy and diseased subjects, respectively. We refer to the help of DPdensity functions for details regarding parametrization, prior specification, and implementation.

The survival and ROC curves are estimated by using a Monte Carlo approximation to the posterior means $E(G_x|x)$ and $E(G_y|y)$, which is based on MCMC samples from posterior predictive distribution for a future observation. The optimal cut-off point is based on the efficiency test, EFF = TP + TN, and is built on Cohen's kappa as defined in Kraemer (1992).

The ROC curve analysis can be performed from the data directly or from the outputs of the DPdensity function (see, example).

Value

An object of class `DProc` representing the ROC curve analysis based on DP mixture of normals models fit. Generic functions such as `print`, and `plot` have methods to show the results of the fit. The results include the estimated densities, cdf's, and ROC curve.

Author(s)

Alejandro Jara <<atjara@uc.cl>>

References

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Kraemer, H. C. (1992). Evaluating Medical Tests. Sage Publications.

See Also

DPdensity

Examples

```
## Not run:
    ##############################################################
    # Fertility data example:
    # The following are Sperm Deformity Index (SDI) values from
    # semen samples of men in an infertility study. They are
    # divided into a "condition" present group defined as those
    # whose partners achieved pregnancy and "condition" absent
    # where there was no pregnancy.
    #
    # Aziz et al. (1996) Sperm deformity index: a reliable
    # predictor of the outcome of fertilization in vitro.
    # Fertility and Sterility, 66(6):1000-1008.
    #
    ##############################################################

     "pregnancy"<- c(165, 140, 154, 139, 134, 154, 120, 133,
                     150, 146, 140, 114, 128, 131, 116, 128,
                     122, 129, 145, 117, 140, 149, 116, 147,
                     125, 149, 129, 157, 144, 123, 107, 129,
                     152, 164, 134, 120, 148, 151, 149, 138,
                     159, 169, 137, 151, 141, 145, 135, 135,
                     153, 125, 159, 148, 142, 130, 111, 140,
                     136, 142, 139, 137, 187, 154, 151, 149,
                     148, 157, 159, 143, 124, 141, 114, 136,
                     110, 129, 145, 132, 125, 149, 146, 138,
                     151, 147, 154, 147, 158, 156, 156, 128,
                     151, 138, 193, 131, 127, 129, 120, 159,
                     147, 159, 156, 143, 149, 160, 126, 136,
                     150, 136, 151, 140, 145, 140, 134, 140,
                     138, 144, 140, 140)

     "nopregnancy"<-c(159, 136, 149, 156, 191, 169, 194, 182,
                      163, 152, 145, 176, 122, 141, 172, 162,
                      165, 184, 239, 178, 178, 164, 185, 154,
                      164, 140, 207, 214, 165, 183, 218, 142,
                      161, 168, 181, 162, 166, 150, 205, 163,
                 166, 176)


########################################################
# Estimating the ROC curve from the data
########################################################

# Initial state

  statex <- NULL
  statey <- NULL

# Prior information

  priorx <-list(alpha=10,m2=rep(0,1),
                s2=diag(100000,1),
                psiinv2=solve(diag(5,1)),
                nu1=6,nu2=4,
                tau1=1,tau2=100)

  priory <-list(alpha=20,m2=rep(0,1),
                s2=diag(100000,1),
                psiinv2=solve(diag(2,1)),
                nu1=6,nu2=4,
                tau1=1,tau2=100)

# MCMC parameters

  nburn<-1000
  nsave<-2000
  nskip<-0
  ndisplay<-100

  mcmcx <- list(nburn=nburn,nsave=nsave,nskip=nskip,
                ndisplay=ndisplay)
  mcmcy <- mcmcx

# Estimating the ROC

  fit1<-DProc(x=pregnancy,y=nopregnancy,priorx=priorx,priory=priory,
              mcmcx=mcmcx,mcmcy=mcmcy,statex=statex,statey=statey,
              statusx=TRUE,statusy=TRUE)
  fit1
  plot(fit1)


########################################################
# Estimating the ROC curve from DPdensity objects
########################################################

  fitx<-DPdensity(y=pregnancy,prior=priorx,mcmc=mcmcx,
                 state=statex,status=TRUE)
```

```
    fity<-DPdensity(y=nopregnancy,prior=priory,mcmc=mcmcy,
                    state=statey,status=TRUE)

  # Estimating the ROC

    fit2<-DProc(fitx=fitx,fity=fity)

    fit2
    plot(fit2)


## End(Not run)
```

DPsurvint *Bayesian analysis for a semiparametric AFT regression model*

Description

This function generates a posterior density sample from a semiparametric AFT regression model for interval-censored data.

Usage

```
DPsurvint(formula,prior,mcmc,state,status,
          data=sys.frame(sys.parent()),na.action=na.fail)
```

Arguments

formula	a two-sided linear formula object describing the model fit, with the response on the left of a ~ operator and the terms, separated by + operators, on the right. In the response matrix, the unknown limits should be -999.
prior	a list giving the prior information. The list includes the following parameter: a0 and b0 giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, alpha giving the value of the precision parameter (it must be specified if a0 and b0 are missing, see details below), m0 and s0 giving the mean and variance of the normal prior distribution for the mean of the log normal baseline distribution, and, tau1 and tau2 giving the hyperparameters for the prior distribution of the variance of the log normal baseline distribution, and beta0 and Sbeta0 giving the hyperparameters of the normal prior distribution for the regression coefficients.
mcmc	a list giving the MCMC parameters. The list must include the following integers: nburn giving the number of burn-in scans, nskip giving the thinning interval, nsave giving the total number of scans to be saved, ndisplay giving the number of saved scans to be displayed on the screen (the function reports on the screen when every ndisplay iterations have been carried out), and tune giving the Metropolis tuning parameter.

`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `DPsurvint` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a Mixture of Dirichlet process in a AFT regression model for interval censored data (Hanson and Johnson, 2004):

$$T_i = exp(-X_i\beta)V_i, i = 1, \ldots, n$$

$$\beta|\beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

$$V_i|G \sim G$$

$$G|\alpha, G_0 \sim DP(\alpha G_0)$$

where, $G_0 = LogNormal(V|\mu, \sigma)$. To complete the model specification, independent hyperpriors are assumed,

$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\mu|m_0, s_0 \sim N(m_0, s_0)$$

$$\sigma^{-1}|\tau_1, \tau_2 \sim Gamma(\tau_1/2, \tau_2/2)$$

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

In the computational implementation of the model, G is considered as latent data and sampled partially with sufficient accuracy to be able to generate $V_1, \ldots, V_{n+1}$ which are exactly iid G, as proposed by Doss (1994). Both Ferguson's definition of DP and the Sethuraman-Tiwari (1982) representation of the process are used, as described in Hanson and Johnson (2004) to allow the inclusion of covariates.

A Metropolis-Hastings step is used to sample the fully conditional distribution of the regression coefficients and errors (see, Hanson and Johnson, 2004). An extra step which moves the clusters in such a way that the posterior distribution is still a stationary distribution, is performed in order to improve the rate of mixing.

Value

An object of class DPsurvint representing the semiparametric AFT regression model fit. Generic functions such as print, plot, summary, and anova have methods to show the results of the fit. The results include beta, mu, sigma, the precision parameter alpha, and the number of clusters.

The function predict.DPsurvint can be used to extract posterior information of the survival curve.

The list state in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set status=TRUE and create the list state based on this starting values. In this case the list state must include the following objects:

beta	giving the value of the regression coefficients.
v	giving the value of the errors (it must be consistent with the data.
mu	giving the mean of the lognormal baseline distribution.
sigma	giving the variance of the lognormal baseline distribution.
alpha	giving the value of the precision parameter.

Author(s)

Alejandro Jara <<atjara@uc.cl>>

References

Doss, H. (1994). Bayesian nonparametric estimation for incomplete data using mixtures of Dirichlet priors. The Annals of Statistics, 22: 1763 - 1786.

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Hanson, T., and Johnson, W. (2004) A Bayesian Semiparametric AFT Model for Interval-Censored Data. Journal of Computational and Graphical Statistics, 13: 341-361.

Sethuraman, J., and Tiwari, R. C. (1982) Convergence of Dirichlet Measures and the Interpretation of their Parameter, in Statistical Decision Theory and Related Topics III (vol. 2), eds. S. S. Gupta and J. O. Berger, New York: Academic Press, pp. 305 - 315.

See Also

predict.DPsurvint

Examples

```
## Not run:
    ####################################
    # A simulated Data Set
    ####################################
     ind<-rbinom(100,1,0.5)
     vsim<-ind*rnorm(100,1,0.25)+(1-ind)*rnorm(100,3,0.25)
     x1<-rep(c(0,1),50)
     x2<-rnorm(100,0,1)
     etasim<-x1+-1*x2
```

```
 time<-vsim*exp(-etasim)
 y<-matrix(-999,nrow=100,ncol=2)
 for(i in 1:100){
    for(j in 1:15){
     if((j-1)<time[i] & time[i]<=j){
        y[i,1]<-j-1
        y[i,2]<-j
     }
 }
 if(time[i]>15)y[i,1]<-15
 }

# Initial state
  state <- NULL

# MCMC parameters

  nburn<-20000
  nsave<-10000
  nskip<-10
  ndisplay<-100
  mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,
               ndisplay=ndisplay,tune=0.125)

# Prior information
  prior <- list(alpha=1,beta0=rep(0,2),Sbeta0=diag(1000,2),
                m0=0,s0=1,tau1=0.01,tau2=0.01)


# Fit the model

  fit1 <- DPsurvint(y~x1+x2,prior=prior,mcmc=mcmc,
                    state=state,status=TRUE)
  fit1

# Summary with HPD and Credibility intervals
  summary(fit1)
  summary(fit1,hpd=FALSE)

# Plot model parameters
# (to see the plots gradually set ask=TRUE)
  plot(fit1,ask=FALSE)
  plot(fit1,ask=FALSE,nfigr=2,nfigc=2)

# Plot an specific model parameter
# (to see the plots gradually set ask=TRUE)
  plot(fit1,ask=FALSE,nfigr=1,nfigc=2,param="x1")
  plot(fit1,ask=FALSE,nfigr=1,nfigc=2,param="mu")

# Table of Pseudo Contour Probabilities
  anova(fit1)

# Predictive information with covariates
```

```
  npred<-10
  xnew<-cbind(rep(1,npred),seq(-1.5,1.5,length=npred))
  xnew<-rbind(xnew,cbind(rep(0,npred),seq(-1.5,1.5,length=npred)))
  grid<-seq(0.00001,14,0.5)
  pred1<-predict(fit1,xnew=xnew,grid=grid)

# Plot Baseline information
  plot(pred1,all=FALSE,band=TRUE)


##############################################################
# Time to Cosmetic Deterioration of Breast Cancer Patients
##############################################################

  data(deterioration)
  attach(deterioration)
  y<-cbind(left,right)

# Initial state
  state <- NULL

# MCMC parameters

  nburn<-20000
  nsave<-10000
  nskip<-20
  ndisplay<-1000
  mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,
               ndisplay=ndisplay,tune=0.25)

# Prior information
  prior <- list(alpha=10,beta0=rep(0,1),Sbeta0=diag(100,1),
                m0=0,s0=1,tau1=0.01,tau2=0.01)

# Fitting the model

  fit2 <- DPsurvint(y~trt,prior=prior,mcmc=mcmc,
                    state=state,status=TRUE)
  fit2

# Summary with HPD and Credibility intervals
  summary(fit2)
  summary(fit2,hpd=FALSE)

# Plot model parameters
# (to see the plots gradually set ask=TRUE)
  plot(fit2)

# Table of Pseudo Contour Probabilities
  anova(fit2)

# Predictive information with covariates
  xnew<-matrix(c(0,1),nrow=2,ncol=1)
```

```
        grid<-seq(0.01,70,1)
        pred2<-predict(fit2,xnew=xnew,grid=grid)
        plot(pred2,all=FALSE,band=TRUE)


## End(Not run)
```

fleabeetles *Flea-beetles*

Description

This data set consider physical information of 74 male flea-beetles reported by Lubischew (1962). Information of three species (Ch. concinna, Ch. heptapotamica, and Ch. heikertingeri) is considered and 6 measurements on each flea-beetles.

Usage

```
data(fleabeetles)
```

Format

A data frame with 74 observations on the following 7 variables.

- `fjft` a numeric vector giving the width of the first joint of the first tarsus in microns (the sum of measurements for both tarsi)
- `sjft` a numeric vector giving the width of the second joint of the first tarsus in microns (the sum of measurements for both tarsi)
- `mwhbee` a numeric vector giving the maximal width of the head between the external edges of the eyes in 0.01 mm
- `mwafp` a numeric vector giving the maximal width of the aedeagus in the fore-part in microns
- `faa` a numeric vector giving the front angle of the aedeagus (1 unit = 7.5 degrees)
- `awfs` a numeric vector giving the aedeagus width from the side in microns
- `species` a numeric vector giving the species: 1=Ch. concinna, 2= Ch. heptapotamica, and 3=Ch. heikertingeri

Source

Lubischew, A. A. (1962) On the Use of Discriminant Functions in Taxonomy, Biometrics, 18: 455-477.

References

MacEachern, S.N., and Muller, P. (1998) Estimating Mixture of Dirichlet Process Models, Journal of Computational and Graphical Statistics, 7: 223-238.

Examples

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

`FPTbinary` *Bayesian analysis for a Finite Polya Tree Bernoulli regression model*

Description

This function generates a posterior density sample for a binary regression model using a Finite Polya tree prior for the link function.

Usage

```
FPTbinary(formula,baseline="logistic",prior,mcmc,state,
          status,misc=NULL,data=sys.frame(sys.parent()),
          na.action=na.fail)
```

Arguments

`formula`	a two-sided linear formula object describing the model fit, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right.
`baseline`	a description of the baseline error distribution to be used in the model. The baseline distributions considered by `FPTbinary` so far is `logistic`.
`prior`	a list giving the prior information. The list includes the following parameters: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Finite Polya Tree prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` and `b0` are missing), `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the regression coefficients, `M` giving the finite level to be considered for the Finite Polya tree.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, `ndisplay` giving the number of saved scans to be displayed on the screen (the function reports on the screen when every `ndisplay` iterations have been carried out), and `tune1` and `tune2` giving the Metropolis tuning parameters for the regression coefficients and precision parameter, respectively (the default value is 1.1).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.

`misc`	misclassification information. When used, this list must include two objects, `sens` and `spec`, giving the sensitivity and specificity, respectively. Both can be a vector or a scalar. This information is used to correct for misclassification in the conditional bernoulli model.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `FPTbinary` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a semiparametric binary regression model using a Finite Polya tree prior (FPT) for the link function (see, Hanson, 2006; Jara, Garcia-Zattera and Lesaffre, 2006):

$$y_i = I(V_i \leq X_i\beta), i = 1, \ldots, n$$

$$V_1, \ldots, V_n|G \sim G$$

$$G|\alpha \sim FPT^M(\Pi, A)$$

where, the FPT is centered around a $Logistic(0, 1)$ distribution if the baseline is `logistic`, by taking each m level of the partition Π to coincide with the $k/2^m, k = 0, \ldots, 2^m$ quantile of the $Logistic(0, 1$ distribution. The family $A = \{\alpha_e : e \in E^*\}$, where $E^* = \bigcup_{m=1}^{M} E^m$ and E^m is the m-fold product of $E = \{0, 1\}$, is specified as $\alpha_{e_1 \ldots e_m} = \alpha m^2$. To complete the model specification, independent hyperpriors are assumed,

$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\beta|\beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

The precision parameter, α, of the `FPT` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

In the computational implementation of the model, Metropolis-Hastings steps are used to sample the posterior distribution of the regression coefficients and the precision parameter, as described in Hanson (2006), and Jara, Garcia-Zattera and Lesaffre (2006).

Value

An object of class `FPTbinary` representing the semiparametric logistic regression model fit. Generic functions such as `print`, `plot`, `predict`, `summary`, and `anova` have methods to show the results of the fit. The results include `beta`, the precision parameter (`alpha`), and the `link` function.

The MCMC samples of the parameters and the errors in the model are stored in the object `thetasave` and `randsave`, respectively. Both objects are included in the list `save.state` and are matrices which can be analyzed directly by functions provided by the coda package.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`beta`	giving the value of the regression coefficients.
`v`	giving the value of the errors (it must be consistent with `yi = I(Vi < xi beta)`.
,	
`y`	giving the value of the true response binary variable (only if the model considers correction for misclassification).
`alpha`	giving the value of the precision parameter.

Author(s)

Alejandro Jara <`<atjara@uc.cl>`>

Tim Hanson <`<hansont@stat.sc.edu>`>

References

Hanson, T. (2006) Inference for Mixtures of Finite Polya tree models. Journal of the American Statistical Association, 101: 1548-1565.

Jara, A., Garcia-Zattera, M.J., Lesaffre, E. (2006) Semiparametric Bayesian Analysis of Misclassified Binary Data. XXIII International Biometric Conference, July 16-21, Montreal, Canada.

Lavine, M. (1992) Some aspects of Polya tree distributions for statistical modelling. The Annals of Statistics, 20: 1222-11235.

Lavine, M. (1994) More aspects of Polya tree distributions for statistical modelling. The Annals of Statistics, 22: 1161-1176.

Examples

```
## Not run:

    # Prostate cancer data example
      data(nodal)
      attach(nodal)
      lacid<-log(acid)

    # Initial state
      state <- NULL

    # MCMC parameters
      nburn<-20000
      nsave<-10000
      nskip<-10
      ndisplay<-100
      mcmc <- list(nburn=nburn,nsave=nsave,
                   nskip=nskip,ndisplay=ndisplay,
                   tune1=1.1,tune2=1.1)

    # Prior distribution
      prior <- list(alpha=1, beta0=c(0,rep(0.75,5)),
                    Sbeta0=diag(c(100,rep(25,5)),6),M=5)
    # Fitting the Finite Polya tree model
      fit1 <- FPTbinary(ssln~age+lacid+xray+size+grade,
                        prior=prior,mcmc=mcmc,
                        state=state,status=TRUE)
      fit1

    # Summary with HPD and Credibility intervals
      summary(fit1)
      summary(fit1,hpd=FALSE)

    # Plot model parameters (to see the plots gradually set ask=TRUE)
      plot(fit1)
      plot(fit1,nfigr=2,nfigc=2)

    # Plot an specific model parameter (to see the plots gradually
    # set ask=TRUE)
      plot(fit1,ask=FALSE,nfigr=1,nfigc=2,param="xray")
      plot(fit1,ask=FALSE,param="link",nfigc=1,nfigr=1)

    # Table of Pseudo Contour Probabilities
      anova(fit1)


    # Fitting parametric models

      nburn<-20000
      nsave<-10000
      nskip<-10
      ndisplay<-100
      mcmc <- list(nburn=nburn,nsave=nsave,
                   nskip=nskip,ndisplay=ndisplay,
                   tune=1.1)

      fit2 <- Pbinary(ssln~age+lacid+xray+size+grade,link="probit",
                      prior=prior,mcmc=mcmc,state=state,status=TRUE)

      fit3 <- Pbinary(ssln~age+lacid+xray+size+grade,link="logit",
                      prior=prior,mcmc=mcmc,state=state,status=TRUE)


    # Model comparison

      DPpsBF(fit1,fit2,fit3)


## End(Not run)
```

FPTrasch *Bayesian analysis for a Finite Polya Tree Rasch model*

Description

This function generates a posterior density sample for a Rasch model, using a Finite Polya Tree or a Mixture of Finite Polya Trees prior for the distribution of the abilities.

Usage

```
FPTrasch(y,prior,mcmc,offset,state,status,
         grid=seq(-10,10,length=1000),
         data=sys.frame(sys.parent()),compute.band=FALSE)
```

Arguments

`y`	a matrix giving the data for which the Rasch Model is to be fitted.
`prior`	a list giving the prior information. The list includes the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Finite Polya tree prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing), `mub` and `Sb` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `tau1` and `tau2` giving the hyperparameters for the prior distribution of variance of the normal baseline distribution, `sigma` giving the standard deviation of the normal baseline distribution (is must be specified if `tau1` and `tau2` are missing), `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the difficulty parameters, and `M` giving the finite level to be considered for the Finite Polya tree.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`offset`	this can be used to specify an a priori known component to the linear predictor.
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`grid`	grid points where the density estimate is evaluated. The default is seq(-10,10,length=1000).
`data`	data frame.
`compute.band`	logical variable indicating whether the confidence band for the density and CDF must be computed.

Details

This generic function fits a Finite Polya Tree Rasch model as in San Martin et al. (2011), where the linear predictor is modeled as follows:

$$\eta_{ij} = \theta_i - \beta_j, i = 1, \ldots, n, j = 1, \ldots, k$$

$$\theta_i | G \sim G$$

$$G|\alpha, \mu, \sigma^2 \sim FPT^M(\Pi^{\mu,\sigma^2}, A)$$

where, the the PT is centered around a $N(\mu, \sigma^2)$ distribution, by taking each m level of the partition Π^{μ,σ^2} to coincide with the $k/2^m, k = 0, \ldots, 2^m$ quantile of the $N(\mu, \sigma^2)$ distribution. The family $A = \{\alpha_e : e \in E^*\}$, where $E^* = \bigcup_{m=0}^{\infty} E^m$ and E^m is the m-fold product of $E = \{0, 1\}$, was specified as $\alpha_{e_1 \ldots e_m} = \alpha m^2$.

To complete the model specification, independent hyperpriors are assumed,

$$\alpha | a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\beta | \beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

$$\mu | \mu_b, S_b \sim N(\mu_b, S_b)$$

$$\sigma^{-2} | \tau_1, \tau_2 \sim Gamma(\tau_1/2, \tau_2/2)$$

Each of the parameters of the baseline distribution, μ and σ can be considered as random or fixed at some particular value. In the first case, a Mixture of Polya Trees Process is considered as a prior for the distribution of the random effects. To let σ to be fixed at a particular value, set τ_1 to NULL in the prior specification. To let μ to be fixed at a particular value, set μ_b to NULL in the prior specification.

In the computational implementation of the model, a Metropolis-Hastings step is used to sample the full conditional of the difficulty parameters. The full conditionals for abilities and PT parameters are sampled using slice sampling. We refer to Jara, Hanson and Lesaffre (2009) for more details and for the description regarding sampling functionals of PTs.

Value

An object of class `FPTrasch` representing the Rasch model fit. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `beta`, `mu`, `sigma2`, and the precision parameter `alpha`.

The function `DPrandom` can be used to extract the posterior mean of the random effects.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`alpha`	giving the value of the precision parameter.
`b`	a vector of dimension nsubjects giving the value of the random effects for each subject.
`beta`	giving the value of the difficulty parameters.
`mu`	giving the mean of the normal baseline distributions.
`sigma2`	giving the variance of the normal baseline distributions.

Author(s)

Alejandro Jara <`<atjara@uc.cl>`>

References

Hanson, T., Johnson, W. (2002) Modeling regression error with a Mixture of Polya Trees. Journal of the American Statistical Association, 97: 1020 - 1033.

Jara, A., Hanson, T., Lesaffre, E. (2009) Robustifying Generalized Linear Mixed Models using a New Class of Mixture of Multivariate Polya Trees. Journal of Computational and Graphical Statistics, 18(4): 838-860.

Lavine, M. (1992) Some aspects of Polya tree distributions for statistical modelling. The Annals of Statistics, 20: 1222-11235.

Lavine, M. (1994) More aspects of Polya tree distributions for statistical modelling. The Annals of Statistics, 22: 1161-1176.

San Martin, E., Jara, A., Rolin, J.-M., and Mouchart, M. (2011) On the Bayesian nonparametric generalization of IRT-type models. Psychometrika (To appear).

See Also

DPrandom, DPrasch

Examples

```
## Not run:
    ####################################
    # A simulated Data Set
    ####################################
      nsubject <- 200
      nitem <- 40

      y <- matrix(0,nrow=nsubject,ncol=nitem)
      dimnames(y) <- list(paste("id",seq(1:nsubject)),
                          paste("item",seq(1,nitem)))


      ind <- rbinom(nsubject,1,0.5)
      theta <- ind*rnorm(nsubject,1,0.25)+(1-ind)*rnorm(nsubject,3,0.25)
      beta <- c(0,seq(-1,3,length=nitem-1))
      true.density <- function(grid)
      {
         0.5*dnorm(grid,1,0.25)+0.5*dnorm(grid,3,0.25)
      }

      for(i in 1:nsubject)
      {
         for(j in 1:nitem)
         {
            eta<-theta[i]-beta[j]
            mean<-exp(eta)/(1+exp(eta))
            y[i,j]<-rbinom(1,1,mean)
         }
      }

    # Prior information
```

```
  beta0 <- rep(0,nitem-1)
  Sbeta0 <- diag(1000,nitem-1)

  prior <- list(alpha=1,
                tau1=6.01,
                tau2=2.01,
                mub=0,
                Sb=100,
                beta0=beta0,
                Sbeta0=Sbeta0,
                M=5)

# Initial state
  state <- NULL

# MCMC parameters

  nburn <- 5000
  nsave <- 5000
  nskip <- 0
  ndisplay <- 100
  mcmc <- list(nburn=nburn,
               nsave=nsave,
               nskip=nskip,
               ndisplay=ndisplay)

# Fit the model
  fit1 <- FPTrasch(y=y,prior=prior,mcmc=mcmc,
                   state=state,status=TRUE,grid=seq(-1,5,0.01),
                   compute.band=TRUE)

# Density estimate (along with HPD band) and truth
  plot(fit1$grid,fit1$dens.u,lwd=2,col="blue",type="l",lty=2,
       xlab=expression(theta),ylab="density")
  lines(fit1$grid,fit1$dens,lwd=2,col="blue")
  lines(fit1$grid,fit1$dens.l,lwd=2,col="blue",lty=2)
  lines(fit1$grid,true.density(fit1$grid),col="red")

# Summary with HPD and Credibility intervals
  summary(fit1)
  summary(fit1,hpd=FALSE)

# Plot model parameters
# (to see the plots gradually set ask=TRUE)
  plot(fit1,ask=FALSE)
  plot(fit1,ask=FALSE,nfigr=2,nfigc=2)

# Extract random effects

  DPrandom(fit1)
  plot(DPrandom(fit1))
  DPcaterpillar(DPrandom(fit1))
```

```
## End(Not run)
```

`FPTraschpoisson` *Bayesian analysis for a Finite Polya Tree Rasch Poisson model*

Description

This function generates a posterior density sample for a Rasch Poisson model, using a Finite Polya Tree or a Mixture of Finite Polya Tree prior for the distribution of the random effects.

Usage

```
FPTraschpoisson(y,prior,mcmc,offset,state,status,
                grid=seq(-10,10,length=1000),data=sys.frame(sys.parent()),
                compute.band=FALSE)
```

Arguments

`y`	a matrix giving the data for which the Rasch Poisson Model is to be fitted.
`prior`	a list giving the prior information. The list includes the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Finite Polya tree prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing), `mub` and `Sb` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `mu` giving the mean of the normal baseline distribution (is must be specified if `mub` and `Sb` are missing), `tau1` and `tau2` giving the hyperparameters for the prior distribution of variance of the normal baseline distribution, `sigma` giving the standard deviation of the normal baseline distribution (is must be specified if `tau1` and `tau2` are missing), `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the difficulty parameters, and `M` giving the finite level to be considered for the Finite Polya tree.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`offset`	this can be used to specify an a priori known component to be included in the linear predictor during the fitting.
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.

`grid`	grid points where the density estimate is evaluated. The default is seq(-10,10,length=1000).
`data`	data frame.
`compute.band`	logical variable indicating whether the confidence band for the density and CDF must be computed.

Details

This generic function fits a semiparametric Rasch Poisson model as in San Martin et al. (2011), where the linear predictor is modeled as follows:

$$\eta_{ij} = \theta_i - \beta_j, i = 1, \ldots, n, j = 1, \ldots, k$$

$$\theta_i | G \sim G$$

$$G | \alpha, \mu, \sigma^2 \sim FPT^M(\Pi^{\mu,\sigma^2}, A)$$

where, the the PT is centered around a $N(\mu, \sigma^2)$ distribution, by taking each m level of the partition Π^{μ,σ^2} to coincide with the $k/2^m, k = 0, \ldots, 2^m$ quantile of the $N(\mu, \sigma^2)$ distribution. The family $A = \{\alpha_e : e \in E^*\}$, where $E^* = \bigcup_{m=0}^{\infty} E^m$ and E^m is the m-fold product of $E = \{0, 1\}$, was specified as $\alpha_{e_1 \ldots e_m} = \alpha m^2$.

To complete the model specification, independent hyperpriors are assumed,

$$\alpha | a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\beta | \beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

$$\mu | \mu_b, S_b \sim N(\mu_b, S_b)$$

$$\sigma^{-2} | \tau_1, \tau_2 \sim Gamma(\tau_1/2, \tau_2/2)$$

Each of the parameters of the baseline distribution, μ and σ can be considered as random or fixed at some particular value. In the first case, a Mixture of Polya Trees Process is considered as a prior for the distribution of the random effects. To let σ^2 to be fixed at a particular value, set τ_1 to NULL in the prior specification. To let μ to be fixed at a particular value, set μ_b to NULL in the prior specification.

In the computational implementation of the model, a Metropolis-Hastings step is used to sample the full conditional of the difficulty parameters. The full conditionals for abilities and PT parameters are sampled using slice sampling. We refer to Jara, Hanson and Lesaffre (2009) for more details and for the description regarding sampling functionals of PTs.

Value

An object of class `FPTraschpoisson` representing the Rasch Poisson model fit. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `beta`, `mu`, `sigma2`, and the precision parameter `alpha`.

The function `DPrandom` can be used to extract the posterior mean of the random effects.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

alpha	giving the value of the precision parameter.
b	a vector of dimension nsubjects giving the value of the random effects for each subject.
beta	giving the value of the difficulty parameters.
mu	giving the mean of the normal baseline distributions.
sigma2	giving the variance of the normal baseline distributions.

Author(s)

Alejandro Jara <<atjara@uc.cl>>

References

Hanson, T., Johnson, W. (2002) Modeling regression error with a Mixture of Polya Trees. Journal of the American Statistical Association, 97: 1020 - 1033.

Jara, A., Hanson, T., Lesaffre, E. (2009) Robustifying Generalized Linear Mixed Models using a New Class of Mixture of Multivariate Polya Trees. Journal of Computational and Graphical Statistics, 18(4): 838-860.

Lavine, M. (1992) Some aspects of Polya tree distributions for statistical modelling. The Annals of Statistics, 20: 1222-11235.

Lavine, M. (1994) More aspects of Polya tree distributions for statistical modelling. The Annals of Statistics, 22: 1161-1176.

San Martin, E., Jara, A., Rolin, J.-M., and Mouchart, M. (2011) On the Bayesian nonparametric generalization of IRT-type models. Psychometrika (To appear).

See Also

DPrandom, DPraschpoisson

Examples

```
## Not run: 
    ####################################
    # A simulated Data Set
    ####################################
      nsubject <- 200
      nitem <- 10

      y <- matrix(0,nrow=nsubject,ncol=nitem)

      ind <- rbinom(nsubject,1,0.5)
      theta <- ind*rnorm(nsubject,1,0.25)+(1-ind)*rnorm(nsubject,3,0.25)
      beta <- c(0,seq(-1,1,length=nitem-1))
      true.density <- function(grid)
      {
         0.5*dnorm(grid,1,0.25)+0.5*dnorm(grid,3,0.25)
      }
      for(i in 1:nsubject)
      {
         for(j in 1:nitem)
         {
            eta<-theta[i]-beta[j]
            mean<-exp(eta)
            y[i,j]<-rpois(1,mean)
         }
      }

    # Prior information

      beta0 <- rep(0,nitem-1)
      Sbeta0 <- diag(1000,nitem-1)

      prior <- list(alpha=1,
                    tau1=6.01,
                    tau2=2.01,
                    mub=0,
                    Sb=100,
                    beta0=beta0,
                    Sbeta0=Sbeta0,
                    M=5)

    # Initial state
      state <- NULL

    # MCMC parameters

      nburn <- 5000
      nsave <- 5000
      nskip <- 0
      ndisplay <- 100
      mcmc <- list(nburn=nburn,
                   nsave=nsave,
                   nskip=nskip,
                   ndisplay=ndisplay)

    # Fit the model
      fit1 <- FPTraschpoisson(y=y,prior=prior,mcmc=mcmc,
                              state=state,status=TRUE,
                              grid=seq(-1,5,0.01),
                              compute.band=TRUE)

    # Density estimate (along with HPD band) and truth
      plot(fit1$grid,fit1$dens.u,lwd=2,col="blue",type="l",lty=2,
           xlab=expression(theta),ylab="density")
      lines(fit1$grid,fit1$dens,lwd=2,col="blue")
      lines(fit1$grid,fit1$dens.l,lwd=2,col="blue",lty=2)
      lines(fit1$grid,true.density(fit1$grid),col="red")

    # Summary with HPD and Credibility intervals
      summary(fit1)
      summary(fit1,hpd=FALSE)

    # Plot model parameters
    # (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE)
      plot(fit1,ask=FALSE,nfigr=2,nfigc=2)

    # Extract random effects
      DPrandom(fit1)
      plot(DPrandom(fit1))
      DPcaterpillar(DPrandom(fit1))


## End(Not run)
```

`fractionation` *British Institute of Radiology Fractionation Studies*

Description

The British Institute of Radiology (BIR) conducted two large-scale randomized clinical trials to assess the effectiveness of different radiotherapy treatment schedules for cancer of the larynx and pharynx. The cambined data come from 858 subjects with laryngeal squamous cell carcinomas and no involvement of asjacent organs. These data have been described and analyzed by Rezvani, Fowler, Hopewell, and Alcock (1993).

Usage

```
data(fractionation)
```

Format

A data frame with 858 observations on the following 6 variables.

`response` Three-year local control 1 (0 no control).

`dose` Total dose (grays).

`df` Total dose x dose/fraction.

`time` Total time of treatment (days).

`kt2` Tumor status (indicators for 2nd level of factor).

`kt3` Tumor status (indicators for 3rd level of factor).

Details

Three-year local control - meaning no detection of laryngeal carcinoma within three years after treatment - is the binary response, coded as 1 if local control is achieved and 0 otherwise. For this data set, three-year local control is achieved for 69 a total dose of radiation is administered in fractions over a treatment period. The dose per fraction df is measured in grays (Gy), the length of treatment period time is measured in days, and the number of fractions of the dose is nf. Tumors are classified by stage (i.e., the extent of invasion), into three groups. This categorical covariate is coded by two indicator variables kt2 and kt3, which are defined by kt2=1 (kt3=1) is the tumor is stage II (stage III) and zero otherwise.

Chappell, Nondahl and Fowler (1995) argued that the tumor stage, the total dose, the total time, and the interaction of the total dose per fraction are the relevant explanantory variables affecting probability of local control.

References

Chappell, R., Nondahl, D.M., and Fowler, J.F. (1995) Modelling Dose and Local Control in Radio-theraphy. Journal of the American Statistical Asso- ciation, 90: 829 - 838.

Newton, M.A., Czado, C., and Chappell, R. (1996) Bayesian inference for semiparametric binary regression. Journal of the American Statistical Association, 91, 142-153.

Rezvani, M., Fowler, J., Hopewell, J., and Alcock, C. (1993) Sensitivity of Human Squamous Cell Carcinoma of the Larynx to Fractionated Radiotherapy. British Journal of Radiology, 66: 245 - 255.

Examples

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

`galaxy` *Galaxy velocities*

Description

This data set consider physical information on velocities (km/second) for 82 galaxies reported by Roeder (1990). These are drawn from six well-separated conic sections of the Corona Borealis region.

Usage

```
data(galaxy)
```

Format

A data frame with 82 observations on the following variable.

`speed` a numeric vector giving the speed of galaxies ((km/second))

Source

Roeder, K. (1990) Density estimation with confidence sets exemplified by superclusters and voids in the galaxies, Journal of the American Statistical Association, 85: 617-624.

References

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Examples

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

`HDPMcdensity` *Bayesian analysis for a hierarchical Dirichlet Process mixture of normals model for conditional density estimation*

Description

This function generates a posterior density sample for a DP mixture of normals model for related random probability measures. Support provided by the NIH/NCI R01CA75981 grant.

Usage

```
HDPMcdensity(formula,study,xpred,ngrid=100,prior,mcmc,
             state,status,data=sys.frame(sys.parent()),
             na.action=na.fail,work.dir=NULL)
```

Arguments

`formula`	a two-sided linear formula object describing the model fit, with the response on the left of a ~ operator and the terms, separated by + operators, on the right.
`study`	a (1 by `nrec`) vector of study indicators. The i-th index is the study i that response j belongs to.
`ngrid`	integer giving the number of grid points where the density estimate is evaluated. The default is 100.
`xpred`	a matrix giving the covariate values where the predictive density is evaluated.
`prior`	a list giving the prior information. The list includes the following parameters: `pe1` and `pe0` giving the prior weights for the point mass at $\epsilon = 1$ and at $\epsilon = 1$, respectively, `ae` and `be` giving the prior parameters for a Beta prior on ϵ, `eps` giving the value of ϵ (it must be specified if `pe1` is missing), `a0` and `b0` vectors giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the vector of precision parameters (it must be specified if `a0` is missing), `m0` and `S0` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `mub` giving the mean of the normal baseline distribution (is must be specified if `m0` is missing), `nub` and `tbinv` giving the hyperparameters of the inverse Wishart prior distribution for the variance of the normal baseline distribution, `sigmab` giving the variance of the normal baseline distribution (is must be specified if `nub` is missing), `nu` and `tinv` giving the hyperparameters of the inverse Wishart

	prior distribution for the variance of the normal kernel, and `sigma` giving the covariance matrix of the normal kernel (is must be specified if `nu` is missing).
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, `ndisplay` giving the number of saved scans to be displayed on screen.
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis (not available yet).
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state` (not available yet).
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `HDPdensity` to print an error message and terminate if there are any incomplete observations.
`work.dir`	working directory.

Details

This generic function fits a hierarchical mixture of DPM of normals model for conditional density estimation (Mueller, Quintana and Rosner, 2004). Let $d_i = (y_i, x_i)$ be the vector of full data, including the responses y_i and predictors x_i. The `HDPMcdensity` function fits the hierarchical mixture of DPM of normals model to the full data and then look at the implied conditional distribution of the responses given the predictors. The model is given by:

$$d_{ij}|F_i \sim F_i$$

where, d_{ij} denote the j-th observation in the i-th study, i=1,...,I, and F_i is assumed to arise as a mixture $F_i = \epsilon H_0 + (1-\epsilon)H_i$ of one common distribution H_0 and a distribution H_i that is specific or idiosyncratic to the i-th study.

The random probability measures H_i in turn are given a Dirichlet process mixture of normal prior. We assume

$$H_i(d) = \int N(\mu, \Sigma) dG_i(\mu),\ i = 0, 1, \ldots, I$$

with

$$G_i|\alpha_i, G_0 \sim DP(\alpha G_0)$$

where, the baseline distribution is

$$G_0 = N(\mu|\mu_b, \Sigma_b)$$

.

To complete the model specification, independent hyperpriors are assumed (optional),

$$\Sigma|\nu, T \sim IW(\nu, T)$$

$$\alpha_i|a_{0i}, b_{0i} \sim Gamma(a_{0i}, b_{0i})$$

$$\mu_b|m_0, S_0 \sim N(m_0, S_0)$$

$$\Sigma_b|\nu_b, Tb \sim IW(\nu_b, Tb)$$

and

$$p(\epsilon) = \pi_0\delta_0 + \pi_1\delta_1 + (1 - \pi_0 - \ pi_1)Be(a_\epsilon, b_\epsilon)$$

Note that the inverted-Wishart prior is parametrized such that if $A \sim IW_q(\nu, \psi)$ then $E(A) = \psi^{-1}/(\nu - q - 1)$.

Value

An object of class `HDPMcdensity` representing the hierarchical DPM of normals model. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `sigma`, `eps`, the vector of precision parameters `alpha`, `mub` and `sigmab`.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`ncluster`	an integer giving the number of clusters.
`ss`	an interger vector defining to which of the `ncluster` clusters each subject belongs.
`sc`	an integer vector defining to which DP each cluster belongs. Note that length(sc)=nrec (only the first `ncluster` elements are considered to start the chain.
`alpha`	giving the vector of dimension nsuties+1 of precision parameters.
`muclus`	a matrix of dimension (number of subject + 100) times the total number of variables (responses + predictors), giving the means for each cluster (only the first `ncluster` rows are considered to start the chain).
`mub`	giving the mean of the normal baseline distributions.
`sigmab`	giving the covariance matrix the normal baseline distributions.
`sigma`	giving the normal kernel covariance matrix.
`eps`	giving the value of `eps`.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

Peter Mueller <<`pmueller@mdanderson.org`>>

References

Mueller, P., Quintana, F. and Rosner, G. (2004). A Method for Combining Inference over Related Nonparametric Bayesian Models. Journal of the Royal Statistical Society, Series B, 66: 735-749.

See Also

`predict.HDPMcdensity`

Examples

```
## Not run:
    # Data
      data(calgb)
      attach(calgb)
      y <- cbind(Z1,Z2,Z3,T1,T2,B0,B1)
      x <- cbind(CTX,GM,AMOF)

      z <- cbind(y,x)

    #  Data for prediction
      data(calgb.pred)
      xpred <- as.matrix(calgb.pred[,8:10])


    # Prior information
      prior <- list(pe1=0.1,
                    pe0=0.1,
                    ae=1,
                    be=1,
                    a0=rep(1,3),
                    b0=rep(1,3),
                    nu=12,
                    tinv=0.25*var(z),
    m0=apply(z,2,mean),
                    S0=var(z),
    nub=12,
                    tbinv=var(z))


    # Initial state
      state <- NULL

    # MCMC parameters

      mcmc <- list(nburn=5000,
                   nsave=5000,
                   nskip=3,
                   ndisplay=100)

    # Fitting the model
      fit1 <- HDPMcdensity(formula=y~x,
                          study=~study,
                          xpred=xpred,
                          prior=prior,
                          mcmc=mcmc,
                          state=state,
                          status=TRUE)

    # Posterior inference
      fit1
      summary(fit1)
    # Plot the parameters
    # (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE)

    # Plot the a specific parameters
    # (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE,param="eps",nfigr=1,nfigc=2)

    # Plot the measure for each study
    # under first values for the predictors, xpred[1,]
      predict(fit1,pred=1,i=1,r=1) # pred1, study 1
      predict(fit1,pred=1,i=2,r=1) # pred1, study 2

    # Plot the measure for each study
    # under second values for the predictors, xpred[2,]
      predict(fit1,pred=2,i=1,r=1) # pred2, study 1
      predict(fit1,pred=2,i=2,r=1) # pred2, study 2

    # Plot the idiosyncratic measure for each study
    # under first values for the predictors, xpred[1,]
      predict(fit1,pred=1,i=1,r=0) # study 1
      predict(fit1,pred=1,i=2,r=0) # study 2

    # Plot the common measure
    # under first values for the predictors, xpred[1,]
      predict(fit1,pred=1,i=0)

## End(Not run)
```

HDPMdensity — *Bayesian analysis for a hierarchical Dirichlet Process mixture of normals model for marginal density estimation*

Description

This function generates a posterior density sample for a DP mixture of normals model for related random probability measures. Support provided by the NIH/NCI R01CA75981 grant.

Usage

```
HDPMdensity(y,study,ngrid=100,prior,mcmc,state,
            status,data=sys.frame(sys.parent()),
            na.action=na.fail,work.dir=NULL)
```

Arguments

`y`	a matrix of responses.
`study`	a (1 by `nrec`) vector of study indicators. The i-th index is the study i that response j belongs to.

`ngrid`	integer giving the number of grid points where the density estimate is evaluated. The default is 100.
`prior`	a list giving the prior information. The list includes the following parameters: `pe1` and `pe0` giving the prior weights for the point mass at $\epsilon = 1$ and at $\epsilon = 1$, respectively, `ae` and `be` giving the prior parameters for a Beta prior on ϵ, `eps` giving the value of ϵ (it must be specified if `pe1` is missing), `a0` and `b0` vectors giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the vector of precision parameters (it must be specified if `a0` is missing), `m0` and `S0` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `mub` giving the mean of the normal baseline distribution (is must be specified if `m0` is missing), `nub` and `tbinv` giving the hyperparameters of the inverse Wishart prior distribution for the variance of the normal baseline distribution, `sigmab` giving the variance of the normal baseline distribution (is must be specified if `nub` is missing), `nu` and `tinv` giving the hyperparameters of the inverse Wishart prior distribution for the variance of the normal kernel, and `sigma` giving the covariance matrix of the normal kernel (is must be specified if `nu` is missing).
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, `ndisplay` giving the number of saved scans to be displayed on screen.
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis (not available yet).
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state` (not available yet).
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `HDPdensity` to print an error message and terminate if there are any incomplete observations.
`work.dir`	working directory.

Details

This generic function fits a hierarchical mixture of DPM of normals model for density estimation (Mueller, Quintana and Rosner, 2004):

$$y_{ij}|F_i \sim F_i$$

where, y_{ij} denote the j-th observation in the i-th study, i=1,...,I, and F_i is assumed to arise as a mixture $F_i = \epsilon H_0 + (1-\epsilon)H_i$ of one common distribution H_0 and a distribution H_i that is specific or idiosyncratic to the i-th study.

The random probability measures H_i in turn are given a Dirichlet process mixture of normal prior. We assume

$$H_i(y) = \int N(\mu, \Sigma) dG_i(\mu),\ i = 0, 1, \ldots, I$$

with

$$G_i|\alpha_i, G_0 \sim DP(\alpha G_0)$$

where, the baseline distribution is

$$G_0 = N(\mu|\mu_b, \Sigma_b)$$

.

To complete the model specification, independent hyperpriors are assumed (optional),

$$\Sigma|\nu, T \sim IW(\nu, T)$$

$$\alpha_i|a_{0i}, b_{0i} \sim Gamma(a_{0i}, b_{0i})$$

$$\mu_b|m_0, S_0 \sim N(m_0, S_0)$$

$$\Sigma_b|\nu_b, Tb \sim IW(\nu_b, Tb)$$

and

$$p(\epsilon) = \pi_0\delta_0 + \pi_1\delta_1 + (1 - \pi_0 - \ pi_1)Be(a_\epsilon, b_\epsilon)$$

Note that the inverted-Wishart prior is parametrized such that if $A \sim IW_q(\nu, \psi)$ then $E(A) = \psi^{-1}/(\nu - q - 1)$.

Value

An object of class `HDPMdensity` representing the hierarchical DPM of normals model. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `sigma`, `eps`, the vector of precision parameters `alpha`, `mub` and `sigmab`.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`ncluster`	an integer giving the number of clusters.
`ss`	an interger vector defining to which of the `ncluster` clusters each subject belongs.
`sc`	an integer vector defining to which DP each cluster belongs. Note that length(sc)=nrec (only the first `ncluster` elements are considered to start the chain.
`alpha`	giving the vector of dimension nsuties+1 of precision parameters.
`muclus`	a matrix of dimension (number of subject + 100) times the number of variables, giving the means for each cluster (only the first `ncluster` rows are considered to start the chain).
`mub`	giving the mean of the normal baseline distributions.
`sigmab`	giving the covariance matrix the normal baseline distributions.
`sigma`	giving the normal kernel covariance matrix.
`eps`	giving the value of `eps`.

Author(s)

Alejandro Jara <<atjara@uc.cl>>

Peter Mueller <<pmueller@mdanderson.org>>

References

Mueller, P., Quintana, F. and Rosner, G. (2004). A Method for Combining Inference over Related Nonparametric Bayesian Models. Journal of the Royal Statistical Society, Series B, 66: 735-749.

See Also

`predict.HDPMdensity`

Examples

```
## Not run:

    # Data
      data(calgb)
      attach(calgb)
      y <- cbind(Z1,Z2,Z3,T1,T2,B0,B1)

    # Prior information
      prior <- list(pe1=0.1,
                    pe0=0.1,
                    ae=1,
                    be=1,
                    a0=rep(1,3),
                    b0=rep(1,3),
                    nu=9,
                    tinv=0.25*var(y),
      m0=apply(y,2,mean),
                    S0=var(y),
      nub=9,
                    tbinv=var(y))


    # Initial state
      state <- NULL

    # MCMC parameters

      mcmc <- list(nburn=5000,
                   nsave=5000,
                   nskip=3,
                   ndisplay=100)

    # Fitting the model
      fit1 <- HDPMdensity(y=y,
                          study=study,
                          prior=prior,
                              mcmc=mcmc,
                              state=state,
                              status=TRUE)

    # Posterior inference
      fit1
      summary(fit1)

    # Plot the parameters
    # (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE)

    # Plot the a specific parameters
    # (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE,param="eps",nfigr=1,nfigc=2)

   # Plot the measure for each study
      predict(fit1,i=1,r=1) # study 1
      predict(fit1,i=2,r=1) # study 2

    # Plot the idiosyncratic measure for each study
      predict(fit1,i=1,r=0) # study 1
      predict(fit1,i=2,r=0) # study 2

    # Plot the common measure
      predict(fit1,i=0)

## End(Not run)
```

`hiv` *HIV-AIDS data*

Description

This data set considers information from a cohort of 262 hemophiliacs at risk of human immunodeficiency virus (HIV) infection from infusions of blood they received periodically to treat their hemophilia in two hospitals in France. All infected patients are believed to have become infected by contaminated blood factor: 105 patients received at least 1,000 micro grams/kg of blood factor for at least one year between 1982 and 1985 (heavily treated group), and 157 patients received less than 1,000 micro grams/kg in each year (lighter treated group). For this cohort both infection with HIV and the onset of acquired immunodeficiency syndrome (AIDS) or other clinical symptoms could be subject to censoring. Therefore, the induction time between infection and clinical AIDS are treated as doubly-censored.

Usage

```
data(hiv)
```

Format

A data frame with 262 observations on the following 5 variables.

`onsetL` a numeric vector giving the lower limit of the HIV infection interval.

`onsetU` a numeric vector giving the upper limit of the HIV infection interval.

`failureL` a numeric vector giving the lower limit of the interval were clinical AIDS was observed.

`failureU` a numeric vector giving the upper limit of the interval were clinical AIDS was observed.

`trt` a numeric vector giving the treatment indicator: (0) indicates the lighter treated group while (1) indicates the heavily treated group.

Details

This dataset was analyzed by dataset De Gruttola and Lagakos (1989). The periodic observation of HIV infection status in these patients was possible because blood samples were stored and retrospectively tested for evidence of infection with the HIV. Note that both the distribution of chronological time of infection and induction time are of interest. In De Gruttola and Lagakos (1989) the proposed nonparametric maximum likelihood one-sample estimator was illustrated by considering the intervals for the onset and failure time, which were the results of a discretization of the time axis into 6-month intervals.

Source

De Grutola, V. and Lagakos, S.W. (1989). Analysis of doubly-censored survival data, with application to AIDS. Biometrics, 45: 1-11.

References

Jara, A., Lesaffre, E., De Iorio, M., Quintana, F. (2010). Bayesian semiparametric inference for multivariate doubly-interval-censored data. Annals of Applied Statistics, 4: 2126-2149.

Examples

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

`igg` *Immunoglobulin G concentrations*

Description

This data set consider information on the serum immunoglbulin G (IgG) concentration from 298 children aged 6 months to 6 years old. The data were reported by Isaacs et al. (1983). These data were further analysed by Royston and Wright (1998) and Kapitula and Bedrick (2005) using the parametric exponential normal family, which includes parameters for skew and kurtosis that can be functions of covariates.

Usage

```
data(igg)
```

Format

A data frame with 298 observations on the following 2 variables.

`age`	a numeric vector giving the age of the children (in years).
`igg`	a numeric vector giving the the serum IgG concentration.

Source

Isaacs, D., Altman, D. G., Tidmarsh, C. E., Valman, H. B., Webster, A. D. B. (1983). Serum immunoglobulin concentrations in preschool children measured by laser nephelometry: reference ranges for IgG, IgA, IgM. J. Clin. Pathol. 36: 1193 - 1196.

References

Kapitula, L. R., Bedrick, E. J. (2005). Diagnostics for the exponential normal growth curve model. Statist. Med. 24: 95 - 108.

Royston, P., Wright, E. M. (1998). A method for estimation age-specific reference intervals ('normal ranges') based on fractional polynomials and exponential transformation. J. R. Statist. Soc. A 161: 79 - 101.

Examples

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

`indon` *Indonesian Children's Health Study*

Description

This data set consider respiratory infection information of 250 indonesian children reported by Sommer, Katz, and Tarwotjo (1984). The children, all preschoolers, were seen quarterly for up to six quarters. At each examination, the presence or absence of respiratory infection was noted.

Usage

```
data(indon)
```

Format

A data frame with 1200 observations on the following 9 variables.

`id` an ordered factor giving a unique identifier for the subject in the study.

`gender` a numeric vector giving the gender.

`height` a numeric vector giving the height for age as a percentage of the National Center for Health Statitics standard centered at 90.

`cosv` a numeric vector giving the seasonal cosine for the annual cycle.

`sinv` a numeric vector giving the seasonal sine for the annual cycle.

`xero` a numeric vector giving the presence (1) or absence (0) of xeropthalmia.

`baseage` a numeric vector giving the age at the entry.

`age` a numeric vector giving the age of the child in months centered at 36

`infect` a numeric vector giving the presence (1) or absence (0) of respiratory infection.

Source

Sommer, A., Katz, J., and Tarwotjo, I. (1984) Increased risk of respiratory infection and diarrhea in children with pre-existing mild vitamin A deficiency, American Journal of Clinical Nutrition, 40: 1090-1095.

References

Zeger, S.L., and Karim, M.R. (1991) Generalized linear models with random effects: A Gibbs sampling approach. Journal of the American Statistical Association, 86: 79-86.

Examples

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

`LDBDPdensity` *Bounded Density Regression using Dependent Bernstein Polynomials*

Description

This function generates a posterior density sample for a Linear Dependent Bernstein-Dirichlet Process model for bounded conditional density estimation.

Usage

```
LDBDPdensity(formula,xpred,prior,mcmc,state,status,ngrid=100,
             grid=NULL,compute.band=FALSE,type.band="PD",
             data=sys.frame(sys.parent()),
             na.action=na.fail,work.dir=NULL)
```

Arguments

`formula`	a two-sided linear formula object describing the model fit, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right. The design matrix is used to model the distribution of the response in the LDBDP model. The response is assumed to take values in [0,1].
`xpred`	a matrix giving the covariate values where the predictive density is evaluated.
`prior`	a list giving the prior information. The list includes the following parameter: `lambda` a double precision giving the parameter of the truncated Poisson prior distribution for the degree, `k`, of the Bernstein polynomial, `maxn` an integer giving the truncation of the the stick-breaking approximation to the dependent Dirichlet process, `alpha` giving the value of the precision parameter of the dependent Dirichlet process, `m0` and `S0` giving the hyperparameters of the normal prior distribution for the mean, `mub`, of the normal baseline distribution `nu` and `psiinv` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix, `Sb`, of the baseline distribution.
`mcmc`	a list giving the MCMC parameters. The list must include the following elements: `nburn` an integer giving the number of burn-in scans, `nskip` an integer giving the thinning interval, `nsave` an integer giving the total number of scans to be saved, `ndisplay` an integer giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out), `slicebeta` a double precision giving the Slice sampling parameter for the regression coefficients, and `slicev` a double precision giving the Slice sampling parameter for the stick-breaking parameters.
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`ngrid`	integer giving the number of grid points where the conditional density estimate is evaluated. The default is 100.
`grid`	vector of grid points where the conditional density estimate is evaluated. The default value is NULL and the grid is chosen according to the range of the data.
`compute.band`	logical variable indicating whether the credible band for the conditional density and mean function must be computed.
`type.band`	string indication the type of credible band to be computed; if equal to "HPD" or "PD" then the 95 percent pointwise HPD or PD band is computed, respectively.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `LDBDPdensity` to print an error message and terminate if there are any incomplete observations.
`work.dir`	working directory.

Details

This generic function fits a Linear Dependent Dirichlet-Bernstein model (Barrientos, Jara and Quintana, 2010), given by:

$$y_i|G_{X_i} \sim G_{X_i}$$

$$\{G_X : X \in \mathcal{X}\}|k, \alpha, G_0 \sim LDBDP(k, \alpha G_0)$$

where, $G_0 = N(\beta|\mu_b, S_b)$. To complete the model specification, independent hyperpriors are assumed,

$$k|\lambda \sim Poisson(\lambda)I(k \geq 1)$$

$$\mu_b|m_0, S_0 \sim N(m_0, S_0)$$

$$S_b|\nu, \Psi \sim IW(\nu, \Psi)$$

Note that the inverted-Wishart prior is parametrized such that if $A \sim IW_q(\nu, \psi)$ then $E(A) = \psi^{-1}/(\nu - q - 1)$.

Note also that the LDBDP model is a extension of the Bernstein-Dirichlet model for density estimation (Petrone, 1999a, 1999b; Petrone and Waserman, 2002).

The computational implementation of the model is based on the finite approximation to the dependent Dirichlet process prior and on the use of conditional MCMC methods. The regression coefficients and stick-breaking parameters are updated jointly using multivariate Slice sampling (Neal, 2003). The degree of the Bernstein polynomial is updated using a Metropolis-Hasting algorithm.

Value

An object of class `LDBDPdensity` representing the LDBDP model fit. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `k`. `mub` and `Sb`.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`k`	an integer giving the degree of the Bernstein polynomial.
`beta`	a matrix of dimension `maxn` times the number of columns in the design matrix, giving the regression coefficients for each stick-breaking component.
`alpha`	giving the value of the precision parameter.
`mub`	giving the mean of the normal baseline distributions.
`Sb`	giving the covariance matrix the normal baseline distributions.
`v`	giving the `maxn` vector of stick-breaking beta random variables. The last element in this vector must be equal to 1.

Author(s)

Felipe Barrientos <<`afbarrie@.mat.puc.cl`>>

Alejandro Jara <<`atjara@uc.cl`>>

References

Barrientos, F., Jara, A., Quintana, F. (2010). Bounded density regression using dependent Bernstein polynomials. Technical Report, Department of Statistics, Pontificia Universidad Catolica de Chile.

Neal, R. (2003) Slice sampling. Anals of Statistics, 31: 705-767.

Petrone, S. (1999a) Random Bernstein polynomials. Scandinavian Journal of Statistics, 26: 373-393.

Petrone, S. (1999b) Bayesian density estimation using Bernstein polynomials. The Canadian Journal of Statistics, 27: 105-126.

Petrone, S. and Waserman, L. (2002) Consistency of Bernstein polynomial posterior. Journal of the Royal Statistical Society, Series B, 64: 79-100.

See Also

`DPcdensity`, `LDDPdensity`

Examples

```
## Not run:

    ########################
    # Simulate data
    ########################
      nrec <- 500
      x <- runif(nrec)
      y <- rep(0,nrec)
      for(i in 1:nrec)
      {
           y[i] <- ifelse(runif(1) < (0.8-0.5*x[i]^2),
                    rbeta(1,22-(x[i]^2)*20,5+x[i]*20),
                    rbeta(1,8+x[i]*5,20))
      }

    # true model

      true.dens <- function(grid,x)
      {
  (0.8-0.5*x^2)*dbeta(grid,22-(x^2)*20,5+x*20)+
          (0.2+0.5*x^2)*dbeta(grid,8+x*5,20)
      }

      true.mean <- function(x)
      {
          (0.8-0.5*x^2)*(22-(x^2)*20)/(22-(x^2)*20+5+x*20)+
          (0.2+0.5*x^2)*(8+x*5)/(8+x*5+20)
      }

    # predictions
      grid <- seq(0,1,len=100)
      npred <- 25
      xpred <- matrix(1,ncol=2,nrow=npred)
      xpred[,2] <- seq(0,1,len=npred)

    # prior
      prior <- list(maxn = 25,
                    alpha = 1,
                    lambda = 25,
                    nu = 4,
                    psiinv = diag(1000,2),
                    m0 = rep(0,2),
                    S0 = diag(1000,2))

    # mcmc
      mcmc <- list(nburn = 5000,
                   nskip = 3,
                   ndisplay = 100,
                   nsave = 5000)

    # state
      state <- NULL

    # fitting the model

      fit <- LDBDPdensity(formula=y~x,xpred=xpred,
                          prior=prior,
                          mcmc=mcmc,
                          state=NULL,status=TRUE,
                          grid=grid,
                          compute.band=TRUE,type.band="PD")

      fit
      summary(fit)
      plot(fit)

    # Plots for some predictions
    # (conditional density and mean function)

      par(mfrow=c(2,2))
      plot(fit$grid,fit$densp.h[7,],type="l",lwd=2,
           xlim=c(0,1),ylim=c(0,4),xlab="y",ylab="density",lty=2)
      lines(fit$grid,fit$densp.m[7,],lwd=2,lty=1)
      lines(fit$grid,fit$densp.l[7,],lwd=2,lty=2)
      lines(fit$grid,true.dens(fit$grid,fit$xpred[7,2]),lwd=2,col="red")

      plot(fit$grid,fit$densp.h[13,],type="l",lwd=2,
           xlim=c(0,1),ylim=c(0,4),xlab="y",ylab="density",lty=2)
      lines(fit$grid,fit$densp.m[13,],lwd=2,lty=1)
      lines(fit$grid,fit$densp.l[13,],lwd=2,lty=2)
      lines(fit$grid,true.dens(fit$grid,fit$xpred[13,2]),lwd=2,col="red")

      plot(fit$grid,fit$densp.h[19,],type="l",lwd=2,
           xlim=c(0,1),ylim=c(0,4),xlab="y",ylab="density",lty=2)
      lines(fit$grid,fit$densp.m[19,],lwd=2,lty=1)
      lines(fit$grid,fit$densp.l[19,],lwd=2,lty=2)
```

```
    lines(fit$grid,true.dens(fit$grid,fit$xpred[19,2]),lwd=2,col="red")

    plot(x,y)
    lines(fit$xpred[,2],fit$meanfp.m,lwd=2,lty=1)
    lines(fit$xpred[,2],fit$meanfp.l,lwd=2,lty=2)
    lines(fit$xpred[,2],fit$meanfp.h,lwd=2,lty=2)
    lines(fit$xpred[,2],true.mean(fit$xpred[,2]),lwd=2,lty=1,col="red")

## End(Not run)
```

LDDPdensity — *Bayesian analysis for a Linear Dependent Dirichlet Process Mixture Model*

Description

This function generates a posterior density sample for a Linear Dependent Dirichlet Process Mixture of Normals model. Support provided by the NIH/NCI R01CA75981 grant.

Usage

```
LDDPdensity(formula,zpred,prior,mcmc,state,status,ngrid=100,
            grid=NULL,compute.band=FALSE,type.band="PD",
            data=sys.frame(sys.parent()),na.action=na.fail,
            work.dir=NULL)
```

Arguments

`formula`	a two-sided linear formula object describing the model fit, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right. The design matrix is used to model the distribution of the response in the LDPP mixture of normals model.
`zpred`	a matrix giving the covariate values where the predictive density is evaluated.
`prior`	a list giving the prior information. The list includes the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing), `m0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `mub` giving the mean of the normal baseline distribution of the regression coefficients (is must be specified if `m0` is missing), `nu` and `psiinv` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix, `sigmab`, of the baseline distribution, `sigmab` giving the variance of the baseline distribution (is must be specified if `nu` is missing), `tau1` giving the hyperparameter for the prior distribution of variance of the normal kernel, and `taus1` and `taus2` giving th hyperparameters of the gamma distribution for `tau2`.

`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`ngrid`	integer giving the number of grid points where the conditional density estimate is evaluated. The default is 100.
`grid`	vector of grid points where the conditional density estimate is evaluated. The default value is NULL and the grid is chosen according to the range of the data.
`compute.band`	logical variable indicating whether the credible band for the conditional density and mean function must be computed.
`type.band`	string indication the type of credible band to be computed; if equal to "HPD" or "PD" then the 95 percent pointwise HPD or PD band is computed, respectively.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `LDDPdensity` to print an error message and terminate if there are any incomplete observations.
`work.dir`	working directory.

Details

This generic function fits a Linear Dependent Dirichlet Process Mixture of Normals model,

$$y_i|f_{X_i} \sim f_{X_i}$$

$$f_{X_i} = \int N(X_i\beta, \sigma^2)G(d\beta d\sigma^2)$$

$$G|\alpha, G_0 \sim DP(\alpha G_0)$$

where, $G_0 = N(\beta|\mu_b, s_b)\Gamma(\sigma^2|\tau_1/2, \tau_2/2)$. To complete the model specification, independent hyperpriors are assumed,

$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\mu_b|m_0, S_{\beta_0} \sim N(m_0, S_{\beta_0})$$

$$s_b|\nu, \Psi \sim IW(\nu, \Psi)$$

$$\tau_2|\tau_{s1}, \tau_{s2} \sim Gamma(\tau_{s1}/2, \tau_{s2}/2)$$

Note that the inverted-Wishart prior is parametrized such that if $A \sim IW_q(\nu, \psi)$ then $E(A) = \psi^{-1}/(\nu - q - 1)$.

Note also that the LDDP model is a natural and simple extension of the the ANOVA DDP model discussed in in De Iorio et al. (2004). The same model is used in Mueller et al.(2005) as the random effects distribution in a repeated measurements model.

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

The computational implementation of the model is based on the marginalization of the `DP` and on the use of MCMC methods for non-conjugate DPM models (see, e.g, MacEachern and Muller, 1998; Neal, 2000).

Value

An object of class `LDDPdensity` representing the LDDP mixture of normals model fit. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `mub`, `sb`, `tau2`, the precision parameter `alpha`, and the number of clusters.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`betaclus`	a matrix of dimension (number of subject + 100) times the number of columns in the design matrix, giving the regression coefficients for each cluster (only the first `ncluster` are considered to start the chain).
`sigmaclus`	a vector of dimension (number of subjects + 100) giving the variance of the normal kernel for each cluster (only the first `ncluster` are considered to start the chain).
`alpha`	giving the value of the precision parameter.
`mub`	giving the mean of the normal baseline distributions.
`sb`	giving the covariance matrix the normal baseline distributions.
`ncluster`	an integer giving the number of clusters.
`ss`	an interger vector defining to which of the `ncluster` clusters each subject belongs.
`tau2`	giving the value of the tau2 parameter.

Author(s)

Alejandro Jara <`<atjara@uc.cl>`>

Peter Mueller <`<pmueller@math.utexas.edu>`>

Gary L. Rosner <`<grosner@jhmi.edu>`>

References

De Iorio, M., Muller, P., Rosner, G., and MacEachern, S. (2004), An ANOVA model for dependent random measures. Journal of the American Statistical Association, 99(465): 205-215.

De Iorio, M., Muller, P., Rosner, G.L., and MacEachern, S (2004) An ANOVA Model for Dependent Random Measures. Journal of the American Statistical Association, 99: 205-215

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

MacEachern, S. N. and Muller, P. (1998) Estimating mixture of Dirichlet Process Models. Journal of Computational and Graphical Statistics, 7 (2): 223-338.

Mueller, P., Rosner, G., De Iorio, M., and MacEachern, S. (2005). A Nonparametric Bayesian Model for Inference in Related Studies. Applied Statistics, 54 (3), 611-626.

Neal, R. M. (2000). Markov Chain sampling methods for Dirichlet process mixture models. Journal of Computational and Graphical Statistics, 9: 249-265.

See Also

DPcdensity

Examples

```
## Not run:

    ##########################################################
    # Simulate data from a mixture of two normal densities
    ##########################################################
      nobs <- 500
      y1   <-rnorm(nobs, 3,.8)

      ## y2 = 0.6
      y21 <- rnorm(nobs,1.5, 0.8)
      y22 <- rnorm(nobs,4.0, 0.6)
      u <- runif(nobs)
      y2 <- ifelse(u<0.6,y21,y22)
      y <- c(y1,y2)

      ## design matrix including a single factor
      trt <- c(rep(0,nobs),rep(1,nobs))

      ## design matrix for posterior predictive
      zpred <- rbind(c(1,0),c(1,1))

    # Prior information

      S0 <- diag(100,2)
      m0 <- rep(0,2)
      psiinv <- diag(1,2)

      prior <- list(a0=10,
                    b0=1,
                    nu=4,
                    m0=m0,
                    S0=S0,
                    psiinv=psiinv,
                    tau1=6.01,
                    taus1=6.01,
                    taus2=2.01)

    # Initial state
      state <- NULL
```

```
# MCMC parameters

  nburn <- 5000
  nsave <- 5000
  nskip <- 3
  ndisplay <- 100
  mcmc <- list(nburn=nburn,
               nsave=nsave,
               nskip=nskip,
               ndisplay=ndisplay)

# Fit the model
  fit1 <- LDDPdensity(y~trt,prior=prior,mcmc=mcmc,
                      state=state,status=TRUE,
                      ngrid=200,zpred=zpred,
                      compute.band=TRUE,type.band="PD")


# Plot posterior density estimate
# with design vector x0=(1,0)

  plot(fit1$grid,fit1$densp.h[1,],type="l",xlab="Y",
       ylab="density",lty=2,lwd=2)
  lines(fit1$grid,fit1$densp.l[1,],lty=2,lwd=2)
  lines(fit1$grid,fit1$densp.m[1,],lty=1,lwd=3)

  # add true density to the plot
  p1 <- dnorm(fit1$grid, 3.0, 0.8)
  lines(fit1$grid,p1,lwd=2,lty=1, col="red")

# Plot posterior density estimate
# with design vector x0=(1,1)

  plot(fit1$grid,fit1$densp.h[2,],type="l",xlab="Y",
       ylab="density",lty=2,lwd=2)
  lines(fit1$grid,fit1$densp.l[2,],lty=2,lwd=2)
  lines(fit1$grid,fit1$densp.m[2,],lty=1,lwd=3)

  # add true density to the plot
  p2 <- 0.6*dnorm(fit1$grid, 1.5, 0.8) +
        0.4*dnorm(fit1$grid, 4.0, 0.6)
  lines(fit1$grid,p2,lwd=2,lty=1, col="red")


# Plot posterior CDF estimate
# with design vector x0=(1,0)

  plot(fit1$grid,fit1$cdfp.h[1,],type="l",xlab="Y",
       ylab="density",lty=2,lwd=2)
  lines(fit1$grid,fit1$cdfp.l[1,],lty=2,lwd=2)
  lines(fit1$grid,fit1$cdfp.m[1,],lty=1,lwd=3)
```

```
    # add true CDF to the plot
    p1 <- pnorm(fit1$grid, 3.0, 0.8)
    lines(fit1$grid,p1,lwd=2,lty=1, col="red")

  # Plot posterior CDF estimate
  # with design vector x0=(1,1)

    plot(fit1$grid,fit1$cdfp.h[2,],type="l",xlab="Y",
         ylab="density",lty=2,lwd=2)
    lines(fit1$grid,fit1$cdfp.l[2,],lty=2,lwd=2)
    lines(fit1$grid,fit1$cdfp.m[2,],lty=1,lwd=3)

    # add true density to the plot
    p2 <- 0.6*pnorm(fit1$grid, 1.5, 0.8) +
          0.4*pnorm(fit1$grid, 4.0, 0.6)
    lines(fit1$grid,p2,lwd=2,lty=1, col="red")


## End(Not run)
```

LDDPrasch *Bayesian analysis for a dependent semiparametric Rasch model*

Description

This function generates a posterior density sample for a semiparametric Rasch model, using a LDDP mixture of normals prior for the distribution of the random effects.

Usage

```
LDDPrasch(formula,prior,mcmc,offset=NULL,
          state,status,
          grid=seq(-10,10,length=1000),
          zpred,data=sys.frame(sys.parent()),
          compute.band=FALSE)
```

Arguments

`formula`	a two-sided linear formula object describing the model fit, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right. The design matrix is used to model the distribution of the response in the LDPP mixture of normals model.
`prior`	a list giving the prior information. The list includes the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing), `m0` and `S0` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline

	distribution, `mub` giving the mean of the normal baseline distribution of the regression coefficients (is must be specified if `m0` is missing), `nu` and `psiinv` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix, `sigmab`, of the baseline distribution, `sigmab` giving the variance of the baseline distribution (is must be specified if `nu` is missing), `tau1` giving the hyperparameter for the prior distribution of variance of the normal kernel, and `taus1` and `taus2` giving th hyperparameters of the gamma distribution for `tau2`, `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the difficulty parameters.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`offset`	this can be used to specify an a priori known component to be included in the linear predictor during the fitting.
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`grid`	grid points where the density estimate is evaluated. The default is seq(-10,10,length=1000).
`zpred`	a matrix giving the covariate values where the predictive density is evaluated.
`data`	data frame.
`compute.band`	logical variable indicating whether the confidence band for the density and CDF must be computed.

Details

This generic function fits a linear dependent semiparametric Rasch model as in Farina et al. (2009), where

$$\eta_{ij} = \theta_i - \beta_j, i = 1, \ldots, n, j = 1, \ldots, k$$

$$\beta|\beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

$$\theta_i|f_{X_i} \sim f_{X_i}$$

$$f_{X_i} = \int N(X_i\alpha_c, \sigma^2)G(d\alpha_c d\sigma^2)$$

$$G|\alpha, G_0 \sim DP(\alpha G_0)$$

where, $G_0 = N(\alpha_c|\mu_b, s_b)\Gamma(\sigma^{-2}|\tau_1/2, \tau_2/2)$. To complete the model specification, the following independent hyperpriors are assumed,

$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\mu_b|m_0, S_0 \sim N(m_0, S_0)$$

$$s_b|\nu, \Psi \sim IW(\nu, \Psi)$$

$$\tau_2|\tau_{s1}, \tau_{s2} \sim Gamma(\tau_{s1}/2, \tau_{s2}/2)$$

Note that the inverted-Wishart prior is parametrized such that if $A \sim IW_q(\nu, \psi)$ then $E(A) = \psi^{-1}/(\nu - q - 1)$.

Note also that the LDDP model is a natural and simple extension of the the ANOVA DDP model discussed in in De Iorio et al. (2004). The same model is used in Mueller et al.(2005) as the random effects distribution in a repeated measurements model.

The precision or total mass parameter, α, of the DP prior can be considered as random, having a gamma distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

The computational implementation of the model is based on the marginalization of the DP and on the use of MCMC methods for non-conjugate DPM models (see, e.g, MacEachern and Muller, 1998; Neal, 2000).

Value

An object of class LDDPrasch representing the LDDP mixture of normals Rasch model. Generic functions such as print, plot, and summary have methods to show the results of the fit. The results include beta, mub, sb, tau2, the precision parameter alpha, and the number of clusters.

The list state in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set status=TRUE and create the list state based on this starting values. In this case the list state must include the following objects:

b	a vector of dimension nsubjects giving the value of the random effects for each subject.
beta	giving the value of the difficulty parameters.
alphaclus	a matrix of dimension (number of subject + 100) times the number of columns in the design matrix, giving the regression coefficients for each cluster (only the first ncluster are considered to start the chain).
sigmaclus	a vector of dimension (number of subjects + 100) giving the variance of the normal kernel for each cluster (only the first ncluster are considered to start the chain).
alpha	giving the value of the precision parameter.
mub	giving the mean of the normal baseline distributions.
sb	giving the covariance matrix the normal baseline distributions.
ncluster	an integer giving the number of clusters.
ss	an interger vector defining to which of the ncluster clusters each subject belongs.
tau2	giving the value of the tau2 parameter.

Author(s)

Alejandro Jara <<atjara@uc.cl>>

References

De Iorio, M., Muller, P., Rosner, G., and MacEachern, S. (2004), An ANOVA model for dependent random measures," Journal of the American Statistical Association, 99(465): 205-215.

De Iorio, M., Johnson, W., Muller, P., and Rosner, G.L. (2009) Bayesian Nonparametric Nonproportional Hazards Survival Modeling. Biometrics, To Appear.

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Farina, P., Quintana, E., San Martin, E., Jara, A. (2009). A Dependent Semiparametric Rasch Model for the Analysis of Chilean Educational Data. In preparation.

MacEachern, S. N. and Muller, P. (1998) Estimating mixture of Dirichlet Process Models. Journal of Computational and Graphical Statistics, 7 (2): 223-338.

Mueller, P., Rosner, G., De Iorio, M., and MacEachern, S. (2005). A Nonparametric Bayesian Model for Inference in Related Studies. Applied Statistics, 54 (3), 611-626.

Neal, R. M. (2000). Markov Chain sampling methods for Dirichlet process mixture models. Journal of Computational and Graphical Statistics, 9: 249-265.

See Also

DPrandom, DPMrasch, DPrasch, FPTrasch

Examples

```
## Not run:
    ####################################
    # A simulated Data Set
    ####################################

      grid <- seq(-4,4,0.01)

      dtrue1 <- function(grid)
      {
         0.6*dnorm(grid,-1,0.4)+
         0.3*dnorm(grid,0,0.5)+
         0.1*dnorm(grid,1,0.5)
      }

      dtrue2 <- function(grid)
      {
         0.5*dnorm(grid,-1,0.5)+
         0.5*dnorm(grid,1,0.5)
      }

      dtrue3 <- function(grid)
      {
         0.1*dnorm(grid,-1,0.5)+
         0.3*dnorm(grid,0,0.5)+
         0.6*dnorm(grid,1,0.4)
      }
```

```
rtrue1 <- function(n)
{
    ind <- sample(x=c(1,2,3),
                  size=n,replace =TRUE,
                  prob =c(0.6,0.3,0.1))
    x1 <- rnorm(n,-1,0.4)
    x2 <- rnorm(n, 0,0.5)
    x3 <- rnorm(n, 1,0.5)
    x <- rep(0,n)
    x[ind==1] <- x1[ind==1]
    x[ind==2] <- x2[ind==2]
    x[ind==3] <- x3[ind==3]
    return(x)
}

rtrue2 <- function(n)
{
    ind <- sample(x=c(1,2),
                  size=n,replace=TRUE,
                  prob =c(0.5,0.5))
    x1 <- rnorm(n,-1,0.5)
    x2 <- rnorm(n, 1,0.5)
    x <- rep(0,n)
    x[ind==1] <- x1[ind==1]
    x[ind==2] <- x2[ind==2]
    return(x)
}

rtrue3 <- function(n)
{
    ind <- sample(x=c(1,2,3),
                  size=n,replace=TRUE,
                  prob =c(0.1,0.3,0.6))
    x1 <- rnorm(n,-1,0.5)
    x2 <- rnorm(n, 0,0.5)
    x3 <- rnorm(n, 1,0.4)
    x <- rep(0,n)
    x[ind==1] <- x1[ind==1]
    x[ind==2] <- x2[ind==2]
    x[ind==3] <- x3[ind==3]
    return(x)
}

b1 <- rtrue1(n=200)
hist(b1,prob=TRUE,xlim=c(-4,4),ylim=c(0,0.7))
lines(grid,dtrue1(grid))

b2 <- rtrue2(n=200)
hist(b2,prob=TRUE,xlim=c(-4,4),ylim=c(0,0.7))
lines(grid,dtrue2(grid))

b3 <- rtrue3(n=200)
hist(b3,prob=TRUE,xlim=c(-4,4),ylim=c(0,0.7))
```

```
   lines(grid,dtrue3(grid))

   nsubject <- 600
   theta <- c(b1,b2,b3)
   trt <- as.factor(c(rep(1,200),rep(2,200),rep(3,200)))
   nitem <- 40

   y <- matrix(0,nrow=nsubject,ncol=nitem)
   dimnames(y)<-list(paste("id",seq(1:nsubject)),
                     paste("item",seq(1,nitem)))

   beta <- c(0,seq(-4,4,length=nitem-1))

   for(i in 1:nsubject)
   {
      for(j in 1:nitem)
      {
         eta <- theta[i]-beta[j]
         prob <- exp(eta)/(1+exp(eta))
         y[i,j] <- rbinom(1,1,prob)
      }
   }

##############################
# design's prediction matrix
##############################

  zpred <- matrix(c(1,0,0,
                    1,1,0,
                    1,0,1),nrow=3,ncol=3,byrow=TRUE)

###########################
# prior
###########################

  prior <- list(alpha=1,
                beta0=rep(0,nitem-1),
                Sbeta0=diag(1000,nitem-1),
                mu0=rep(0,3),
                S0=diag(100,3),
                tau1=6.01,
                taus1=6.01,
                taus2=2.01,
                nu=5,
                psiinv=diag(1,3))

###########################
# mcmc
###########################
  mcmc <- list(nburn=5000,
               nskip=3,
               ndisplay=100,
               nsave=5000)
```

```
###########################
# fitting the model
###########################

  fitLDDP <-  LDDPrasch(formula=y ~ trt,
                        prior=prior,
                        mcmc=mcmc,
                        state=NULL,
                        status=TRUE,
                        zpred=zpred,
                        grid=grid,compute.band=TRUE)

  fitLDDP

  summary(fitLDDP)

#########################################
# plots
#########################################
  plot(fitLDDP)

  plot(fitLDDP,param="prediction")

#########################################
# plot the estimated and true densities
#########################################

  par(cex=1.5,mar=c(4.1, 4.1, 1, 1))
  plot(fitLDDP$grid,fitLDDP$dens.m[1,],xlim=c(-4,4),ylim=c(0,0.8),
       type="l",lty=1,lwd=3,xlab="Ability",ylab="density",col=1)
  lines(fitLDDP$grid,fitLDDP$dens.u[1,],lty=2,lwd=3,col=1)
  lines(fitLDDP$grid,fitLDDP$dens.l[1,],lty=2,lwd=3,col=1)
  lines(grid,dtrue1(grid),lwd=3,col="red",lty=3)

  par(cex=1.5,mar=c(4.1, 4.1, 1, 1))
  plot(fitLDDP$grid,fitLDDP$dens.m[2,],xlim=c(-4,4),ylim=c(0,0.8),
       type="l",lty=1,lwd=3,xlab="Ability",ylab="density",col=1)
  lines(fitLDDP$grid,fitLDDP$dens.u[2,],lty=2,lwd=3,col=1)
  lines(fitLDDP$grid,fitLDDP$dens.l[2,],lty=2,lwd=3,col=1)
  lines(grid,dtrue2(grid),lwd=3,col="red",lty=3)

  par(cex=1.5,mar=c(4.1, 4.1, 1, 1))
  plot(fitLDDP$grid,fitLDDP$dens.m[3,],xlim=c(-4,4),ylim=c(0,0.8),
       type="l",lty=1,lwd=3,xlab="Ability",ylab="density",col=1)
  lines(fitLDDP$grid,fitLDDP$dens.u[3,],lty=2,lwd=3,col=1)
  lines(fitLDDP$grid,fitLDDP$dens.l[3,],lty=2,lwd=3,col=1)
  lines(grid,dtrue3(grid),lwd=3,col="red",lty=3)

#########################################
# Extract random effects
#########################################
  DPrandom(fitLDDP)
```

```
    plot(DPrandom(fitLDDP))
    DPcaterpillar(DPrandom(fitLDDP))


## End(Not run)
```

LDDPraschpoisson *Bayesian analysis for a dependent semiparametric Rasch Poisson model*

Description

This function generates a posterior density sample for a semiparametric Rasch Poisson model, using a LDDP mixture of normals prior for the distribution of the random effects.

Usage

```
LDDPraschpoisson(formula,prior,mcmc,
                 offset=NULL,state,status,
                 grid=seq(-10,10,length=1000),
                 zpred,data=sys.frame(sys.parent()),
                 compute.band=FALSE)
```

Arguments

formula	a two-sided linear formula object describing the model fit, with the response on the left of a ~ operator and the terms, separated by + operators, on the right. The design matrix is used to model the distribution of the response in the LDPP mixture of normals model.
prior	a list giving the prior information. The list includes the following parameter: a0 and b0 giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, alpha giving the value of the precision parameter (it must be specified if a0 is missing), m0 and S0 giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, mub giving the mean of the normal baseline distribution of the regression coefficients (is must be specified if m0 is missing), nu and psiinv giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix, sigmab, of the baseline distribution, sigmab giving the variance of the baseline distribution (is must be specified if nu is missing), tau1 giving the hyperparameter for the prior distribution of variance of the normal kernel, and taus1 and taus2 giving th hyperparameters of the gamma distribution for tau2, beta0 and Sbeta0 giving the hyperparameters of the normal prior distribution for the difficulty parameters.
mcmc	a list giving the MCMC parameters. The list must include the following integers: nburn giving the number of burn-in scans, nskip giving the thinning interval, nsave giving the total number of scans to be saved, and ndisplay giving the number of saved scans to be displayed on screen (the function reports on the screen when every ndisplay iterations have been carried out).

`offset`	this can be used to specify an a priori known component to be included in the linear predictor during the fitting.
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`grid`	grid points where the density estimate is evaluated. The default is seq(-10,10,length=1000).
`zpred`	a matrix giving the covariate values where the predictive density is evaluated.
`data`	data frame.
`compute.band`	logical variable indicating whether the confidence band for the density and CDF must be computed.

Details

This generic function fits a linear dependent semiparametric Rasch Poisson model as in Farina et al. (2009), where

$$\eta_{ij} = \theta_i - \beta_j, i = 1, \ldots, n, j = 1, \ldots, k$$

$$\beta|\beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

$$\theta_i|f_{X_i} \sim f_{X_i}$$

$$f_{X_i} = \int N(X_i\alpha_c, \sigma^2)G(d\alpha_c d\sigma^2)$$

$$G|\alpha, G_0 \sim DP(\alpha G_0)$$

where, $G_0 = N(\alpha_c|\mu_b, s_b)\Gamma(\sigma^{-2}|\tau_1/2, \tau_2/2)$. To complete the model specification, the following independent hyperpriors are assumed,

$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\mu_b|m_0, S_0 \sim N(m_0, S_0)$$

$$s_b|\nu, \Psi \sim IW(\nu, \Psi)$$

$$\tau_2|\tau_{s1}, \tau_{s2} \sim Gamma(\tau_{s1}/2, \tau_{s2}/2)$$

Note that the inverted-Wishart prior is parametrized such that if $A \sim IW_q(\nu, \psi)$ then $E(A) = \psi^{-1}/(\nu - q - 1)$.

Note also that the LDDP model is a natural and simple extension of the the ANOVA DDP model discussed in in De Iorio et al. (2004). The same model is used in Mueller et al.(2005) as the random effects distribution in a repeated measurements model.

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

The computational implementation of the model is based on the marginalization of the `DP` and on the use of MCMC methods for non-conjugate DPM models (see, e.g, MacEachern and Muller, 1998; Neal, 2000).

Value

An object of class `LDDPraschpoisson` representing the LDDP mixture of normals Rasch Poisson model. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `beta`, `mub`, `sb`, `tau2`, the precision parameter `alpha`, and the number of clusters.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`b`	a vector of dimension nsubjects giving the value of the random effects for each subject.
`beta`	giving the value of the difficulty parameters.
`alphaclus`	a matrix of dimension (number of subject + 100) times the number of columns in the design matrix, giving the regression coefficients for each cluster (only the first `ncluster` are considered to start the chain).
`sigmaclus`	a vector of dimension (number of subjects + 100) giving the variance of the normal kernel for each cluster (only the first `ncluster` are considered to start the chain).
`alpha`	giving the value of the precision parameter.
`mub`	giving the mean of the normal baseline distributions.
`sb`	giving the covariance matrix the normal baseline distributions.
`ncluster`	an integer giving the number of clusters.
`ss`	an interger vector defining to which of the `ncluster` clusters each subject belongs.
`tau2`	giving the value of the tau2 parameter.

Author(s)

Alejandro Jara <`<atjara@uc.cl>`>

References

De Iorio, M., Muller, P., Rosner, G., and MacEachern, S. (2004), An ANOVA model for dependent random measures," Journal of the American Statistical Association, 99(465): 205-215.

De Iorio, M., Johnson, W., Muller, P., and Rosner, G.L. (2009) Bayesian Nonparametric Nonproportional Hazards Survival Modeling. Biometrics, To Appear.

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Farina, P., Quintana, E., San Martin, E., Jara, A. (2009). A Dependent Semiparametric Rasch Model for the Analysis of Chilean Educational Data. In preparation.

MacEachern, S. N. and Muller, P. (1998) Estimating mixture of Dirichlet Process Models. Journal of Computational and Graphical Statistics, 7 (2): 223-338.

Mueller, P., Rosner, G., De Iorio, M., and MacEachern, S. (2005). A Nonparametric Bayesian Model for Inference in Related Studies. Applied Statistics, 54 (3), 611-626.

Neal, R. M. (2000). Markov Chain sampling methods for Dirichlet process mixture models. Journal of Computational and Graphical Statistics, 9: 249-265.

See Also

DPrandom, DPMraschpoisson, DPraschpoisson, FPTraschpoisson

Examples

```
## Not run:
    ####################################
    # A simulated Data Set
    ####################################

      grid <- seq(-4,4,0.01)

      dtrue1 <- function(grid)
      {
         0.6*dnorm(grid,-1,0.4)+
         0.3*dnorm(grid,0,0.5)+
         0.1*dnorm(grid,1,0.5)
      }

      dtrue2 <- function(grid)
      {
         0.5*dnorm(grid,-1,0.5)+
         0.5*dnorm(grid,1,0.5)
      }

      dtrue3 <- function(grid)
      {
         0.1*dnorm(grid,-1,0.5)+
         0.3*dnorm(grid,0,0.5)+
         0.6*dnorm(grid,1,0.4)
      }

      rtrue1 <- function(n)
      {
          ind <- sample(x=c(1,2,3),
                        size=n,replace=TRUE,
                        prob=c(0.6,0.3,0.1))
          x1 <- rnorm(n,-1,0.4)
          x2 <- rnorm(n, 0,0.5)
          x3 <- rnorm(n, 1,0.5)
          x <- rep(0,n)
          x[ind==1] <- x1[ind==1]
          x[ind==2] <- x2[ind==2]
          x[ind==3] <- x3[ind==3]
          return(x)
      }
```

```
rtrue2 <- function(n)
{
    ind <- sample(x=c(1,2),
                  size=n,replace=TRUE,
                  prob=c(0.5,0.5))
    x1 <- rnorm(n,-1,0.5)
    x2 <- rnorm(n, 1,0.5)
    x <- rep(0,n)
    x[ind==1] <- x1[ind==1]
    x[ind==2] <- x2[ind==2]
    return(x)
}

rtrue3 <- function(n)
{
    ind <- sample(x=c(1,2,3),
                  size=n,replace=TRUE,
                  prob=c(0.1,0.3,0.6))
    x1 <- rnorm(n,-1,0.5)
    x2 <- rnorm(n, 0,0.5)
    x3 <- rnorm(n, 1,0.4)
    x <- rep(0,n)
    x[ind==1] <- x1[ind==1]
    x[ind==2] <- x2[ind==2]
    x[ind==3] <- x3[ind==3]
    return(x)
}

b1 <- rtrue1(n=200)
hist(b1,prob=TRUE,xlim=c(-4,4),ylim=c(0,0.7))
lines(grid,dtrue1(grid))

b2 <- rtrue2(n=200)
hist(b2,prob=TRUE,xlim=c(-4,4),ylim=c(0,0.7))
lines(grid,dtrue2(grid))

b3 <- rtrue3(n=200)
hist(b3,prob=TRUE,xlim=c(-4,4),ylim=c(0,0.7))
lines(grid,dtrue3(grid))

nsubject <- 600
theta <- c(b1,b2,b3)
trt <- as.factor(c(rep(1,200),rep(2,200),rep(3,200)))
nitem <- 5

y <- matrix(0,nrow=nsubject,ncol=nitem)
dimnames(y)<-list(paste("id",seq(1:nsubject)),
                  paste("item",seq(1,nitem)))

beta <- c(0,seq(-3,-1,length=nitem-1))

for(i in 1:nsubject)
```

```
    {
       for(j in 1:nitem)
       {
          eta <- theta[i]-beta[j]
          mm <- exp(eta)
          y[i,j] <- rpois(1,mm)
       }
    }

##############################
# design's prediction matrix
##############################

  zpred <- matrix(c(1,0,0,
                    1,1,0,
                    1,0,1),nrow=3,ncol=3,byrow=TRUE)

###########################
# prior
###########################

  prior <- list(alpha=1,
                beta0=rep(0,nitem-1),
                Sbeta0=diag(1000,nitem-1),
                mu0=rep(0,3),
                S0=diag(100,3),
                tau1=6.01,
                taus1=6.01,
                taus2=2.01,
                nu=5,
                psiinv=diag(1,3))

###########################
# mcmc
###########################
  mcmc <- list(nburn=5000,
               nskip=3,
               ndisplay=100,
               nsave=5000)

###########################
# fitting the model
###########################

  fitLDDP <-  LDDPraschpoisson(formula=y ~ trt,
                               prior=prior,
                               mcmc=mcmc,
                               state=NULL,
                               status=TRUE,
                               zpred=zpred,
                               grid=grid,compute.band=TRUE)

  fitLDDP
```

```
      summary(fitLDDP)

    #########################################
    # plots
    #########################################
      plot(fitLDDP)

      plot(fitLDDP,param="prediction")

    #########################################
    # plot the estimated and true densities
    #########################################

      par(cex=1.5,mar=c(4.1, 4.1, 1, 1))
      plot(fitLDDP$grid,fitLDDP$dens.m[1,],xlim=c(-4,4),ylim=c(0,0.8),
           type="l",lty=1,lwd=3,xlab="Ability",ylab="density",col=1)
      lines(fitLDDP$grid,fitLDDP$dens.u[1,],lty=2,lwd=3,col=1)
      lines(fitLDDP$grid,fitLDDP$dens.l[1,],lty=2,lwd=3,col=1)
      lines(grid,dtrue1(grid),lwd=3,col="red",lty=3)

      par(cex=1.5,mar=c(4.1, 4.1, 1, 1))
      plot(fitLDDP$grid,fitLDDP$dens.m[2,],xlim=c(-4,4),ylim=c(0,0.8),
           type="l",lty=1,lwd=3,xlab="Ability",ylab="density",col=1)
      lines(fitLDDP$grid,fitLDDP$dens.u[2,],lty=2,lwd=3,col=1)
      lines(fitLDDP$grid,fitLDDP$dens.l[2,],lty=2,lwd=3,col=1)
      lines(grid,dtrue2(grid),lwd=3,col="red",lty=3)

      par(cex=1.5,mar=c(4.1, 4.1, 1, 1))
      plot(fitLDDP$grid,fitLDDP$dens.m[3,],xlim=c(-4,4),ylim=c(0,0.8),
           type="l",lty=1,lwd=3,xlab="Ability",ylab="density",col=1)
      lines(fitLDDP$grid,fitLDDP$dens.u[3,],lty=2,lwd=3,col=1)
      lines(fitLDDP$grid,fitLDDP$dens.l[3,],lty=2,lwd=3,col=1)
      lines(grid,dtrue3(grid),lwd=3,col="red",lty=3)

    #########################################
    # Extract random effects
    #########################################
      DPrandom(fitLDDP)
      plot(DPrandom(fitLDDP))
      DPcaterpillar(DPrandom(fitLDPP))


 ## End(Not run)
```

LDDProc *Linear dependent DP model for conditional ROC curve estimation.*

Description

This function generates a posterior density sample for a Linear Dependent Dirichlet Process Mixture of Normals model for conditional ROC curve estimations.

Usage

```
LDDProc(y.d,z.d,y.nond,z.nond,
        zpred.d,zpred.nond=NULL,prior.d,prior.nond=NULL,
        mcmc,state,status,ngrid=100,
        grid=NULL,compute.band=FALSE,type.band="PD",
        data=sys.frame(sys.parent()),na.action=na.fail,
        work.dir=NULL)
```

Arguments

`y.d`	a vector giving the responses for the diseased group.
`z.d`	a matrix giving the design matrix for the diseased group.
`y.nond`	a vector giving the responses for the non-diseased group.
`z.nond`	a matrix giving the design matrix for the non-diseased group.
`zpred.d`	a matrix giving the covariate values where the predictive density is evaluated for the diseased group.
`zpred.nond`	a matrix giving the covariate values where the predictive density is evaluated for the non-diseased group. By default, zpred.nond=NULL which means that zpred.nond=zpred.d.
`prior.d`	a list giving the prior information for the diseased group. The list includes the following parameters: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing), `m0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `mub` giving the mean of the normal baseline distribution of the regression coefficients (is must be specified if `m0` is missing), `nu` and `psiinv` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix, `sigmab`, of the baseline distribution, `sigmab` giving the variance of the baseline distribution (is must be specified if `nu` is missing), `tau1` giving the hyperparameter for the prior distribution of variance of the normal kernel, and `taus1` and `taus2` giving th hyperparameters of the gamma distribution for `tau2`.
`prior.nond`	a list giving the prior information for the non-diseased group. The list includes the same parameters than `prior.d`. The default specification (prior.nond = NULL) uses `prior.nond=prior.d`.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).

`state`	a list giving the current value of the parameters for each individual model. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`ngrid`	integer giving the number of grid points where the conditional density estimates are evaluated. The default is 100.
`grid`	vector of grid points where the conditional density estimate is evaluated. The default value is NULL and the grid is chosen according to the range of the data.
`compute.band`	logical variable indicating whether the credible band for the conditional density and mean function must be computed.
`type.band`	string indication the type of credible band to be computed; if equal to "HPD" or "PD" then the 95 percent pointwise HPD or PD band is computed, respectively.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `LDDProc` to print an error message and terminate if there are any incomplete observations.
`work.dir`	working directory.

Details

The generic function fits the model described in Inacio et al. (2012) for conditional ROC curve estimation. Specifically, the function fits independent Linear Dependent Dirichlet Process Mixture of Normals models for the diseased (i=1) and non-diseased (i=2) groups. The conditional ROC curves are obtained from the conditional densities. The model is given by:

$$y_{ij}|f_{X_{ij}} \sim f_{X_{ij}}$$

$$f_{X_{ij}} = \int N(X_{ij}\beta, \sigma^2) G_i(d\beta d\sigma^2)$$

$$G_i|\alpha_i, G_{0i} \sim DP(\alpha_i G_{0i})$$

where, $G_{0i} = N(\beta|\mu_{bi}, s_{bi})\Gamma(\sigma^2|\tau_1/2, \tau_{2i}/2)$. To complete the model specification, independent hyperpriors are assumed,

$$\alpha_i|a_{0i}, b_{0i} \sim Gamma(a_{0i}, b_{0i})$$

$$\mu_{bi}|m_{0i}, S_{\beta_{0i}} \sim N(m_{0i}, S_{\beta_{0i}})$$

$$s_{bi}|\nu_i, \Psi_i \sim IW(\nu_i, \Psi_i)$$

$$\tau_{2i}|\tau_{s1i}, \tau_{s2i} \sim Gamma(\tau_{s1i}/2, \tau_{s2i}/2)$$

The precision or total mass parameters, α_i, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_{0i}, b_{0i})$, or fixed at some particular value.

We refer the reader to the help file associated with the `LDDPdensity` function for more details about the prior specification, parameterizations and computational strategy.

Value

An object of class `LDDProc` representing the two LDDP mixture of normals model fits. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results for each model include `mub`, `sb`, `tau2`, the precision parameter `alpha`, and the number of clusters.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. Two different objects are included: `state.d` and `state.nd`. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case, each of the lists included in `state` must include the following objects:

`betaclus`	a matrix of dimension (number of subject + 100) times the number of columns in the design matrix, giving the regression coefficients for each cluster (only the first `ncluster` are considered to start the chain).
`sigmaclus`	a vector of dimension (number of subjects + 100) giving the variance of the normal kernel for each cluster (only the first `ncluster` are considered to start the chain).
`alpha`	giving the value of the precision parameter.
`mub`	giving the mean of the normal baseline distributions.
`sb`	giving the covariance matrix the normal baseline distributions.
`ncluster`	an integer giving the number of clusters.
`ss`	an interger vector defining to which of the `ncluster` clusters each subject belongs.
`tau2`	giving the value of the tau2 parameter.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

References

Inacio, V., Jara, A., Hanson, T.E., de Carvalho, M. (2012) Bayesian nonparametric ROC regression modeling with application to diabetes diagnosis. Technical report.

See Also

`LDDPdensity`

Examples

```
## Not run:

    ###############################################################
    # Simulated data example.
    # - Data generated using "perfect" simulation.
    # - one binary predictor.
    # - 250 observations in each
    #   combination of predictor and
    #   status.
    ##############################################################

    # Functions required for simulation

      findq <- function(true.cdf,target,low,upp,
                        epsilon=0.0000001)
      {
           plow <- true.cdf(low)
           pupp <- true.cdf(upp)
           pcenter <- true.cdf((upp+low)/2)
           err <- abs(pcenter-target)
           i <- 0
           while(err > epsilon)
           {
                 i <- i + 1
                 if(target< pcenter)
                 {
                    upp <- (upp+low)/2
                    pupp <- pcenter
                    pcenter <- true.cdf((upp+low)/2)
                    err <- abs(pcenter-target)
                 }
                 if(target>= pcenter)
                 {
                    low <- (upp+low)/2
                    plow <- pcenter
                    pcenter <- true.cdf((upp+low)/2)
                    err <- abs(pcenter-target)
                 }
           }
           return((upp+low)/2)
      }


      true.cdf.nond1 <- function(x)
      {
         pnorm(x,2.1,sqrt(0.0324))
      }

      true.cdf.nond2 <- function(x)
      {
 0.5*pnorm(x,1.85,sqrt(0.005))+
         0.5*pnorm(x,2.25,sqrt(0.005))
      }

      true.cdf.d1 <- function(x)
      {
 0.5*pnorm(x,1.95,sqrt(0.005))+
         0.5*pnorm(x,2.35,sqrt(0.005))
      }

      true.cdf.d2 <- function(x)
      {
          pnorm(x,2.5,sqrt(0.0324))
      }

    # Simulating the data

      nsim <- 250
      qq <- seq(1,nsim)/(nsim+1)

      y.nond1 <- rep(0,nsim)
      for(i in 1:nsim)
      {
          aa <- findq(true.cdf.nond1,qq[i],
                      low=-6,upp=6,epsilon=0.0000001)
          y.nond1[i] <- aa
      }

      y.nond2 <- rep(0,nsim)
      for(i in 1:nsim)
      {
          aa <- findq(true.cdf.nond2,qq[i],
                      low=-6,upp=6,epsilon=0.0000001)
          y.nond2[i] <- aa
      }
      y.nond <- c(y.nond1,y.nond2)
      trt.nond <- c(rep(0,nsim),rep(1,nsim))

      y.d1 <- rep(0,nsim)
      for(i in 1:nsim)
      {
          aa <- findq(true.cdf.d1,qq[i],
                      low=-6,upp=6,epsilon=0.0000001)
          y.d1[i] <- aa
      }

      y.d2 <- rep(0,nsim)
      for(i in 1:nsim)
      {
          aa <- findq(true.cdf.d2,qq[i],
                      low=-6,upp=6,epsilon=0.0000001)
          y.d2[i] <- aa
      }

      y.d <- c(y.d1,y.d2)
      trt.d <- c(rep(0,nsim),rep(1,nsim))

    # Design matrices

      z.d <- cbind(rep(1,2*nsim),trt.d)
      colnames(z.d) <- c("(Intercept)","trt")
      z.nond <- cbind(rep(1,2*nsim),trt.nond)
      colnames(z.nond) <- c("(Intercept)","trt")
```

```
# design matrix for posterior predictive inference

  zpred <- rbind(c(1,0),c(1,1))

# Prior information
  prior <- list(a0=10,
               b0=1,
               nu=4,
               m0=rep(0,2),
               S0=diag(100,2),
               psiinv=diag(1,2),
               tau1=6.01,
               taus1=6.01,
               taus2=2.01)

# Initial state
  state <- NULL

# MCMC parameters

  nburn <- 5000
  nsave <- 5000
  nskip <- 4
  ndisplay <- 500
  mcmc <- list(nburn=nburn,
              nsave=nsave,
              nskip=nskip,
              ndisplay=ndisplay)

# Fitting the model

  fit1 <- LDDProc(y.d=y.d,z.d=z.d,
                 y.nond=y.nond,z.nond=z.nond,
                 zpred.d=zpred,
                 prior.d=prior,
                 prior.nond=prior,
                 mcmc=mcmc,
                 state=state,
                 status=TRUE,
                 compute.band=TRUE)

  fit1
  summary(fit1)
  plot(fit1)


 # Ploting the conditional
 # ROC curve for x=c(1,0),
 # along with the true curve

   par(cex=1.7,mar=c(4.1, 4.1, 1, 1))

   plot(fit1$rocgrid,fit1$rocp.h[1,],type="l",
             lty=2,lwd=2,ylim=c(0,1),xlim=c(0,1),
             xlab="False positive rate",
             ylab="True positive rate")
        lines(fit1$rocgrid,fit1$rocp.l[1,],lty=2,lwd=2)
        lines(fit1$rocgrid,fit1$rocp.m[1,],lty=1,lwd=2)

        nn <- length(fit1$rocgrid)
        tt <- rep(0,nn)
        for(i in 1:nn)
        {
    tt[i] <- findq(true.cdf.nond1,
                          1-fit1$rocgrid[i],
                          low=-6,upp=6,
                          epsilon=0.0000001)
        }
        true.roc1 <- 1.0 - true.cdf.d1(tt)
        lines(fit1$rocgrid,true.roc1,
               lty=1,lwd=3,col="red")

      # Ploting the conditional
      # ROC curve for x=c(1,1),
      # along with the true curve

        par(cex=1.7,mar=c(4.1, 4.1, 1, 1))

        plot(fit1$rocgrid,fit1$rocp.h[2,],type="l",
             lty=2,lwd=2,ylim=c(0,1),xlim=c(0,1),
             xlab="False positive rate",
             ylab="True positive rate")
        lines(fit1$rocgrid,fit1$rocp.l[2,],lty=2,lwd=2)
        lines(fit1$rocgrid,fit1$rocp.m[2,],lty=1,lwd=2)

        nn <- length(fit1$rocgrid)
        tt <- rep(0,nn)
        for(i in 1:nn)
        {
            tt[i] <- findq(true.cdf.nond2,
                          1-fit1$rocgrid[i],
                          low=-6,upp=6,
                          epsilon=0.0000001)
        }
        true.roc2 <- 1.0 - true.cdf.d2(tt)
        lines(fit1$rocgrid,true.roc2,lty=1,lwd=3,col="red")


## End(Not run)
```

`LDDPsurvival` *Bayesian analysis for a Linear Dependent Dirichlet Process Mixture of Survival Models*

Description

This function generates a posterior density sample for a Linear Dependent Dirichlet Process Mixture of Survival models.

Usage

```
LDDPsurvival(formula,zpred,prior,mcmc,
             state,status,grid,
             data=sys.frame(sys.parent()),
             na.action=na.fail,work.dir=NULL)
```

Arguments

`formula`	a two-sided linear formula object describing the model fit, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right. The design matrix is used to model the distribution of the response in the LDPP mixture of normals model. In the response matrix, the unknown limits should be -999.
`zpred`	a matrix giving the covariate values where the predictive density is evaluated.
`prior`	a list giving the prior information. The list includes the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing), `m0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `mub` giving the mean of the normal baseline distribution of the regression coefficients (is must be specified if `m0` is missing), `nu` and `psiinv` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix, `sigmab`, of the baseline distribution, `sigmab` giving the variance of the baseline distribution (is must be specified if `nu` is missing), `tau1` giving the hyperparameter for the prior distribution of variance of the normal kernel, and `taus1` and `taus2` giving th hyperparameters of the gamma distribution for `tau2`.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`grid`	real vector giving the grid points where the density, survival and hazard estimates are evaluated.
`data`	data frame.

`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `LDDPsurvival` to print an error message and terminate if there are any incomplete observations.
`work.dir`	working directory.

Details

This generic function fits a Linear Dependent Dirichlet Process Mixture of Survival models as in De Iorio et al. (2009):

$$T_i \in (a_i, b_i]$$

$$log(T_i) = y_i | f_{X_i} \sim f_{X_i}$$

$$f_{X_i} = \int N(X_i\beta, \sigma^2) G(d\beta d\sigma^2)$$

$$G|\alpha, G_0 \sim DP(\alpha G_0)$$

where, $G_0 = N(\beta|\mu_b, s_b)\Gamma(\sigma^2|\tau_1/2, \tau_2/2)$. To complete the model specification, independent hyperpriors are assumed,

$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\mu_b|m_0, S_0 \sim N(m_0, S_0)$$

$$s_b|\nu, \Psi \sim IW(\nu, \Psi)$$

$$\tau_2|\tau_{s1}, \tau_{s2} \sim Gamma(\tau_{s1}/2, \tau_{s2}/2)$$

Note that the inverted-Wishart prior is parametrized such that if $A \sim IW_q(\nu, \psi)$ then $E(A) = \psi^{-1}/(\nu - q - 1)$.

Note also that the LDDP model is a natural and simple extension of the the ANOVA DDP model discussed in in De Iorio et al. (2004). The same model is used in Mueller et al.(2005) as the random effects distribution in a repeated measurements model.

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

The computational implementation of the model is based on the marginalization of the `DP` and on the use of MCMC methods for non-conjugate DPM models (see, e.g, MacEachern and Muller, 1998; Neal, 2000).

Value

An object of class `LDDPsurvival` representing the LDDP mixture of normals model fit. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `mub`, `sb`, `tau2`, the precision parameter `alpha`, and the number of clusters.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`betaclus`	a matrix of dimension (number of subject + 100) times the number of columns in the design matrix, giving the regression coefficients for each cluster (only the first `ncluster` are considered to start the chain).
`sigmaclus`	a vector of dimension (number of subjects + 100) giving the variance of the normal kernel for each cluster (only the first `ncluster` are considered to start the chain).
`alpha`	giving the value of the precision parameter.
`mub`	giving the mean of the normal baseline distributions.
`sb`	giving the covariance matrix the normal baseline distributions.
`ncluster`	an integer giving the number of clusters.
`ss`	an interger vector defining to which of the `ncluster` clusters each subject belongs.
`tau2`	giving the value of the tau2 parameter.
`y`	giving the value of imputed log-survival times.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

Peter Mueller <<`pmueller@math.utexas.edu`>>

Gary L. Rosner <<`grosner@jhmi.edu`>>

References

De Iorio, M., Muller, P., Rosner, G., and MacEachern, S. (2004) An ANOVA model for dependent random measures. Journal of the American Statistical Association, 99(465): 205-215.

De Iorio, M., Johnson, W., Muller, P., and Rosner, G.L. (2009) Bayesian Nonparametric Nonproportional Hazards Survival Modeling. Biometrics, To Appear.

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

MacEachern, S. N. and Muller, P. (1998) Estimating mixture of Dirichlet Process Models. Journal of Computational and Graphical Statistics, 7 (2): 223-338.

Mueller, P., Rosner, G., De Iorio, M., and MacEachern, S. (2005). A Nonparametric Bayesian Model for Inference in Related Studies. Applied Statistics, 54 (3), 611-626.

Neal, R. M. (2000). Markov Chain sampling methods for Dirichlet process mixture models. Journal of Computational and Graphical Statistics, 9: 249-265.

See Also

`LDDPdensity`

Examples

```
## Not run:

    ##############################################################
    # Time to Cosmetic Deterioration of Breast Cancer Patients
    ##############################################################

      data(deterioration)
      attach(deterioration)
      ymat <- cbind(left,right)

    # design matrix for posterior predictive
      zpred <- rbind(c(1,0),c(1,1))

    # Prior information

      S0 <- diag(100,2)
      m0 <- rep(0,2)
      psiinv <- diag(1,2)

      prior <- list(a0=10,
                    b0=1,
                    nu=4,
                    m0=m0,
                    S0=S0,
                    psiinv=psiinv,
                    tau1=6.01,
                    taus1=6.01,
                    taus2=2.01)

    # Initial state
      state <- NULL

    # MCMC parameters

      nburn <- 5000
      nsave <- 5000
      nskip <- 3
      ndisplay <- 100
      mcmc <- list(nburn=nburn,
                   nsave=nsave,
                   nskip=nskip,
                   ndisplay=ndisplay)

    # Fit the model
      fit1 <- LDDPsurvival(ymat~trt,prior=prior,
                           mcmc=mcmc,state=state,status=TRUE,
                           grid=seq(0.01,70,1),zpred=zpred)


    # Plot posterior density estimates
    # with design vector x0=(1,0)
      plot(fit1$grid,fit1$densp.h[1,],type="l",
           xlab="time",ylab="density",lty=2,lwd=2)
      lines(fit1$grid,fit1$densp.l[1,],lty=2,lwd=2)
      lines(fit1$grid,fit1$densp.m[1,],lty=1,lwd=3)

    # Plot posterior density estimates
    # with design vector x0=(1,1)

      plot(fit1$grid,fit1$densp.h[2,],type="l",
           xlab="time",ylab="density",lty=2,lwd=2)
      lines(fit1$grid,fit1$densp.l[2,],lty=2,lwd=2)
      lines(fit1$grid,fit1$densp.m[2,],lty=1,lwd=3)

    # Plot posterior survival estimates
    # with design vector x0=(1,0)

      plot(fit1$grid,fit1$survp.h[1,],type="l",
           xlab="time",ylab="survival",lty=2,lwd=2,ylim=c(0,1))
      lines(fit1$grid,fit1$survp.l[1,],lty=2,lwd=2)
      lines(fit1$grid,fit1$survp.m[1,],lty=1,lwd=3)

    # Plot posterior survival estimates
    # with design vector x0=(1,1)

      plot(fit1$grid,fit1$survp.h[2,],type="l",
           xlab="time",ylab="survival",lty=2,lwd=2,ylim=c(0,1))
      lines(fit1$grid,fit1$survp.l[2,],lty=2,lwd=2)
      lines(fit1$grid,fit1$survp.m[2,],lty=1,lwd=3)

    # Plot posterior hazard estimates
    # with design vector x0=(1,0)

      plot(fit1$grid,fit1$hazp.h[1,],type="l",
           xlab="time",ylab="hazard",lty=2,lwd=2)
      lines(fit1$grid,fit1$hazp.l[1,],lty=2,lwd=2)
      lines(fit1$grid,fit1$hazp.m[1,],lty=1,lwd=3)

    # Plot posterior hazard estimates
    # with design vector x0=(1,1)

      plot(fit1$grid,fit1$hazp.h[2,],type="l",
           xlab="time",ylab="survival",lty=2,lwd=2)
      lines(fit1$grid,fit1$hazp.l[2,],lty=2,lwd=2)
      lines(fit1$grid,fit1$hazp.m[2,],lty=1,lwd=3)


## End(Not run)
```

`LDDPtwopl` *Bayesian analysis for a dependent semiparametric two parameters logistic model*

Description

This function generates a posterior density sample for a semiparametric two parameters logistic model, using a LDDP mixture of normals prior for the distribution of the random effects.

Usage

```
LDDPtwopl(formula,prior,mcmc,
          state,status,
          grid=seq(-10,10,length=1000),
          zpred,data=sys.frame(sys.parent()),
          compute.band=FALSE)
```

Arguments

`formula`	a two-sided linear formula object describing the model fit, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right. The design matrix is used to model the distribution of the response in the LDPP mixture of normals model.
`prior`	a list giving the prior information. The list includes the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing), `m0` and `S0` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `mub` giving the mean of the normal baseline distribution of the regression coefficients (is must be specified if `m0` is missing), `nu` and `psiinv` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix, `sigmab`, of the baseline distribution, `sigmab` giving the variance of the baseline distribution (is must be specified if `nu` is missing), `tau1` giving the hyperparameter for the prior distribution of variance of the normal kernel, and `taus1` and `taus2` giving th hyperparameters of the gamma distribution for `tau2`, `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the difficulty parameters, and `r1` and `r2` giving the hyperparameters of the lognormal prior distribution for the discrimination parameters.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.

`grid`	grid points where the density estimate is evaluated. The default is seq(-10,10,length=1000).
`zpred`	a matrix giving the covariate values where the predictive density is evaluated.
`data`	data frame.
`compute.band`	logical variable indicating whether the confidence band for the density and CDF must be computed.

Details

This generic function fits a linear dependent semiparametric two parameters logistic model:

$$\eta_{ij} = \gamma_j(\theta_i - \beta_j), i = 1, \ldots, n, j = 1, \ldots, k$$

$$\beta | \beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

$$\gamma_j | r_1, r_2 \sim LN(r_1, r_2), j = 2, \ldots, k$$

$$\theta_i | f_{X_i} \sim f_{X_i}$$

$$f_{X_i} = \int N(X_i \alpha_c, \sigma^2) G(d\alpha_c d\sigma^2)$$

$$G | \alpha, G_0 \sim DP(\alpha G_0)$$

where, $G_0 = N(\alpha_c | \mu_b, s_b)\Gamma(\sigma^{-2} | \tau_1/2, \tau_2/2)$. To complete the model specification, the following independent hyperpriors are assumed,

$$\alpha | a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\mu_b | m_0, S_0 \sim N(m_0, S_0)$$

$$s_b | \nu, \Psi \sim IW(\nu, \Psi)$$

$$\tau_2 | \tau_{s1}, \tau_{s2} \sim Gamma(\tau_{s1}/2, \tau_{s2}/2)$$

Note that the inverted-Wishart prior is parametrized such that if $A \sim IW_q(\nu, \psi)$ then $E(A) = \psi^{-1}/(\nu - q - 1)$.

Note also that the LDDP model is a natural and simple extension of the the ANOVA DDP model discussed in in De Iorio et al. (2004). The same model is used in Mueller et al.(2005) as the random effects distribution in a repeated measurements model.

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

The computational implementation of the model is based on the marginalization of the `DP` and on the use of MCMC methods for non-conjugate DPM models (see, e.g, MacEachern and Muller, 1998; Neal, 2000).

Value

An object of class `LDDPtwopl` representing the LDDP mixture of normals two parameters logistic model. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `beta`, `gamma`, `mub`, `sb`, `tau2`, the precision parameter `alpha`, and the number of clusters.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`b`	a vector of dimension nsubjects giving the value of the random effects for each subject.
`beta`	giving the value of the difficulty parameters.
`gamma`	giving the value of the discrimination parameters.
`alphaclus`	a matrix of dimension (number of subject + 100) times the number of columns in the design matrix, giving the regression coefficients for each cluster (only the first `ncluster` are considered to start the chain).
`sigmaclus`	a vector of dimension (number of subjects + 100) giving the variance of the normal kernel for each cluster (only the first `ncluster` are considered to start the chain).
`alpha`	giving the value of the precision parameter.
`mub`	giving the mean of the normal baseline distributions.
`sb`	giving the covariance matrix the normal baseline distributions.
`ncluster`	an integer giving the number of clusters.
`ss`	an interger vector defining to which of the `ncluster` clusters each subject belongs.
`tau2`	giving the value of the tau2 parameter.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

References

De Iorio, M., Muller, P., Rosner, G., and MacEachern, S. (2004), An ANOVA model for dependent random measures," Journal of the American Statistical Association, 99(465): 205-215.

De Iorio, M., Johnson, W., Muller, P., and Rosner, G.L. (2009) Bayesian Nonparametric Nonproportional Hazards Survival Modeling. Biometrics, To Appear.

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Farina, P., Quintana, E., San Martin, E., Jara, A. (2009). A Dependent Semiparametric Rasch Model for the Analysis of Chilean Educational Data. In preparation.

MacEachern, S. N. and Muller, P. (1998) Estimating mixture of Dirichlet Process Models. Journal of Computational and Graphical Statistics, 7 (2): 223-338.

Mueller, P., Rosner, G., De Iorio, M., and MacEachern, S. (2005). A Nonparametric Bayesian Model for Inference in Related Studies. Applied Statistics, 54 (3), 611-626.

Neal, R. M. (2000). Markov Chain sampling methods for Dirichlet process mixture models. Journal of Computational and Graphical Statistics, 9: 249-265.

See Also

LDDPrasch

Examples

```
## Not run:
    ####################################
    # A simulated Data Set
    ####################################

      grid <- seq(-4,4,0.01)

      dtrue1 <- function(grid)
      {
         0.6*dnorm(grid,-1,0.4)+
         0.3*dnorm(grid,0,0.5)+
         0.1*dnorm(grid,1,0.5)
      }

      dtrue2 <- function(grid)
      {
         0.5*dnorm(grid,-1,0.5)+
         0.5*dnorm(grid,1,0.5)
      }

      dtrue3 <- function(grid)
      {
         0.1*dnorm(grid,-1,0.5)+
         0.3*dnorm(grid,0,0.5)+
         0.6*dnorm(grid,1,0.4)
      }

      rtrue1 <- function(n)
      {
          ind <- sample(x=c(1,2,3),
          size=n,replace =TRUE,
          prob =c(0.6,0.3,0.1))
          x1 <- rnorm(n,-1,0.4)
          x2 <- rnorm(n, 0,0.5)
          x3 <- rnorm(n, 1,0.5)
          x <- rep(0,n)
          x[ind==1] <- x1[ind==1]
          x[ind==2] <- x2[ind==2]
          x[ind==3] <- x3[ind==3]
          return(x)
      }
```

```
rtrue2 <- function(n)
{
    ind <- sample(x=c(1,2),
    size=n,replace=TRUE,
    prob =c(0.5,0.5))
    x1 <- rnorm(n,-1,0.5)
    x2 <- rnorm(n, 1,0.5)
    x <- rep(0,n)
    x[ind==1] <- x1[ind==1]
    x[ind==2] <- x2[ind==2]
    return(x)
}

rtrue3 <- function(n)
{
    ind <- sample(x=c(1,2,3),
    size=n,replace=TRUE,
    prob =c(0.1,0.3,0.6))
    x1 <- rnorm(n,-1,0.5)
    x2 <- rnorm(n, 0,0.5)
    x3 <- rnorm(n, 1,0.4)
    x <- rep(0,n)
    x[ind==1] <- x1[ind==1]
    x[ind==2] <- x2[ind==2]
    x[ind==3] <- x3[ind==3]
    return(x)
}

b1 <- rtrue1(n=200)
hist(b1,prob=TRUE,xlim=c(-4,4),ylim=c(0,0.7))
lines(grid,dtrue1(grid))

b2 <- rtrue2(n=200)
hist(b2,prob=TRUE,xlim=c(-4,4),ylim=c(0,0.7))
lines(grid,dtrue2(grid))

b3 <- rtrue3(n=200)
hist(b3,prob=TRUE,xlim=c(-4,4),ylim=c(0,0.7))
lines(grid,dtrue3(grid))

nsubject <- 600
theta <- c(b1,b2,b3)
trt <- as.factor(c(rep(1,200),rep(2,200),rep(3,200)))
nitem <- 40

y <- matrix(0,nrow=nsubject,ncol=nitem)
dimnames(y)<-list(paste("id",seq(1:nsubject)),
                  paste("item",seq(1,nitem)))

beta <- c(0,sample(seq(-1.8,1.8,length=nitem-1)))
gam <- c(1,sample(seq(0.2,1.4,length=nitem-1)))
```

```
    for(i in 1:nsubject)
    {
       for(j in 1:nitem)
       {
          eta <- gam[j]*(theta[i]-beta[j])
          prob <- exp(eta)/(1+exp(eta))
          y[i,j] <- rbinom(1,1,prob)
       }
    }

##############################
# design's prediction matrix
##############################

  zpred <- matrix(c(1,0,0,
                    1,1,0,
                    1,0,1),nrow=3,ncol=3,byrow=TRUE)

###########################
# prior
###########################

  prior <- list(alpha=1,
                beta0=rep(0,nitem-1),
                Sbeta0=diag(1000,nitem-1),
                r1=0,
                r2=1,
                mu0=rep(0,3),
                S0=diag(100,3),
                tau1=6.01,
                taus1=6.01,
                taus2=2.01,
                nu=5,
                psiinv=diag(1,3))

###########################
# mcmc
###########################
  mcmc <- list(nburn=5000,
               nskip=3,
               ndisplay=200,
               nsave=5000)

###########################
# fitting the model
###########################

  fitLDDP <-  LDDPtwopl(formula=y ~ trt,
                        prior=prior,
                        mcmc=mcmc,
                        state=NULL,
                        status=TRUE,
                        zpred=zpred,
                                 grid=grid,compute.band=TRUE)

      fitLDDP

      summary(fitLDDP)

    #########################################
    # plots
    #########################################
      plot(fitLDDP)

      plot(fitLDDP,param="prediction")

    #########################################
    # plot the estimated and true densities
    #########################################

      par(cex=1.5,mar=c(4.1, 4.1, 1, 1))
      plot(fitLDDP$grid,fitLDDP$dens.m[1,],xlim=c(-4,4),ylim=c(0,0.8),
           type="l",lty=1,lwd=3,xlab="Ability",ylab="density",col=1)
      lines(fitLDDP$grid,fitLDDP$dens.u[1,],lty=2,lwd=3,col=1)
      lines(fitLDDP$grid,fitLDDP$dens.l[1,],lty=2,lwd=3,col=1)
      lines(grid,dtrue1(grid),lwd=3,col="red",lty=3)

      par(cex=1.5,mar=c(4.1, 4.1, 1, 1))
      plot(fitLDDP$grid,fitLDDP$dens.m[2,],xlim=c(-4,4),ylim=c(0,0.8),
           type="l",lty=1,lwd=3,xlab="Ability",ylab="density",col=1)
      lines(fitLDDP$grid,fitLDDP$dens.u[2,],lty=2,lwd=3,col=1)
      lines(fitLDDP$grid,fitLDDP$dens.l[2,],lty=2,lwd=3,col=1)
      lines(grid,dtrue2(grid),lwd=3,col="red",lty=3)

      par(cex=1.5,mar=c(4.1, 4.1, 1, 1))
      plot(fitLDDP$grid,fitLDDP$dens.m[3,],xlim=c(-4,4),ylim=c(0,0.8),
           type="l",lty=1,lwd=3,xlab="Ability",ylab="density",col=1)
      lines(fitLDDP$grid,fitLDDP$dens.u[3,],lty=2,lwd=3,col=1)
      lines(fitLDDP$grid,fitLDDP$dens.l[3,],lty=2,lwd=3,col=1)
      lines(grid,dtrue3(grid),lwd=3,col="red",lty=3)

    #########################################
    # Extract random effects
    #########################################
      DPrandom(fitLDDP)
      plot(DPrandom(fitLDDP))
      DPcaterpillar(DPrandom(fitLDDP))


## End(Not run)
```

`LDPDdoublyint` *Bayesian analysis for a Linear Dependent Poisson Dirichlet Process Mixture Models for the Analysis of Doubly-Interval-Censored Data*

Description

This function generates a posterior density sample for a linear dependent Poisson Dirichlet process mixture model for the analysis of doubly-invertval-censored survival data.

Usage

```
LDPDdoublyint(onset,failure,p,xonset,q,xfailure,xpred,
              grid,prior,mcmc,state,status,work.dir=NULL)
```

Arguments

`onset`	a matrix giving the interval limits for the onset times. For multiple variables the limits must be included in a sequential manner for each variable, i.e., (L1,U1,L2,U2,...).
`failure`	a matrix giving the interval limits for the time-to-event times. For multiple variables the limits must be included in a sequential manner for each variable, i.e. (L1,U1,L2,U2,...).
`p`	an integer giving the number of predictors included for each onset variable. Different design matrices are allowed for the different responses but of the same p-dimension.
`xonset`	a matrix giving the design matrices for each onset time. For multiple variables the design matrices must be included in order and includes the intercepts, i.e. (XO1,XO2,...).
`q`	an integer giving the number of predictors included for each time-to-event variable. Different design matrices are allowed for the different responses but of the same q-dimension.
`xfailure`	a matrix giving the design matrices for each time-to-event variable. For multiple variables the design matrices must be included in order and includes the intercepts, i.e. (XT1,XT2,...).
`xpred`	a matrix giving the value of the predictors for which survival and hazard functions will be evaluated. The number of columns of `xpred` should be (p+q)*nvar/2 where nvar is the number of variables. The design matrices for the predictions must include onset predictors first and then time-to-event predictors, i.e., (XO1,XO2,...,XT1,XT2,...).
`grid`	a matrix including the grids where survival and hazard functions are evaluated. Each row must include the grid points for different variable.
`prior`	a list giving the prior information. The list includes the following parameter: `q`, `a0` and `b0` giving the hyperparameters for prior distribution of the `a` precision parameter of the Poisson-Dirichlet process prior, `mub` and `sigmab` giving the hyperparameters for the prior distributions of the `b` precision parameter of the Poisson-Dirichlet process prior, `nu` and `tinv` giving the hyperparameters of the inverted Wishart prior distribution for the kernel covariance matrix, `mb` and `Sb` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `nub` and `tbinv` giving the hyperparameters of the inverted Wishart prior distribution for the for the covariance matrix of the normal

	baseline distribution, and `maxm` giving the finite truncation limit of the sitck-breaking representation of the Poisson-Dirichlet process.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out), and `tune1` and `tune2` the parameters needed for the MH candidate generating distribution for the precision parameters of the Poisson-Dirichlet process prior.
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`work.dir`	working directory.

Details

This generic function fits a Linear Dependent Poisson-Dirichlet Process Mixture of Survival models as in Jara et al. (2010). The joint distribution of the true chronological onset times and true time-to-events, T_i, is modeled using the mixture model

$$log(T_i) = y_i | f_{X_i} \sim f_{X_i}$$

$$f_{X_i} = \int N(X_i\beta, \Sigma) dG(\beta)$$

$$G|a, b, G_0 \sim PD(a, b, G_0)$$

where, $G_0 = N(\beta|m_b, S_b)$. To complete the model specification, independent hyperpriors are assumed,

$$\Sigma|\nu, T \sim IW(\nu, T)$$

$$a|q, a_0, b_0 \sim q\delta_0 + (1-q)Beta(a_0, b_0)$$

$$b|a, \mu_b, \sigma_b \sim N(\mu_b, \sigma_b)I(-a, \infty)$$

$$m_b|m_0, S_0 \sim N(m_0, S_0)$$

$$S_b|\nu_b, Tb \sim IW(\nu_b, T_b)$$

Note that the inverted-Wishart prior is parametrized such that if $A \sim IW_q(\nu, \psi)$ then $E(A) = \psi^{-1}/(\nu - q - 1)$.

The precision parameters are updated using a MH step. The candidate generating distribution is of the form

$$a'|a, tune_1 \sim 0.5\delta_0 + 0.5N(a, tune_1^2)$$

$$b'|b, a', tune_2 \sim N(b, tune_2^2)I(-a', \infty)$$

The computational implementation of the model is based on the `maxm` truncation of stick-breaking representation of the model (see, Jara et al., 2009).

Value

An object of class `LDPDdoublyint` representing the LDPD mixture of survival models fit. Generic functions such as `print`, `plot`, `summary`, and `predict` have methods to show the results of the fit. The results include `mb`, `Sb`, `sigma`, the precision parameter `alpha`, and the number of clusters.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`alpha`	a vector giving the value of the precision parameters a and b.
`b`	a matrix of dimension maxm times the number of columns in the design matrix ((p+q)*nvar/2), giving the regression coefficients for each cluster.
`sigma`	a matrix of dimension nvar times nvar giving the kernel covariance matrix.
`mb`	giving the mean of the normal baseline distributions.
`Sb`	giving the covariance matrix the normal baseline distributions.
`ncluster`	an integer giving the number of clusters.
`ss`	an interger vector defining to which of the `ncluster` clusters each subject belongs.
`y`	a matrix of dimension nsubject times nvar giving the value of the imputed log-survival times.

Author(s)

Alejandro Jara <`<atjara@uc.cl>`>

References

Jara, A., Lesaffre, E., De Iorio, M., Quintana, F. (2010). Bayesian semiparametric inference for multivariate doubly-interval-censored data. Annals of Applied Statistics, 4: 2126-2149.

See Also

`LDDPdensity`, `LDDPsurvival`

Examples

```
## Not run:

################################################################
# HIV-AIDS Data
################################################################

  data(hiv)
  attach(hiv)

################################################################
# Working folder
```

```
################################################################

  work.dir <- NULL

################################################################
# Response and design matrices
################################################################

  nsubject <- length(onsetL)
  onset <- cbind(onsetL,onsetU)
  failure <- cbind(failureL,failureU)

  intercept <- rep(1,nsubject)
  p <- 2
  xonset <- cbind(IntO=intercept,trtO=trt)
  q <- 2
  xfailure <- cbind(IntF=intercept,trtF=trt)

################################################################
# Predictions
################################################################

   grid <- matrix(c(rep(seq(0,30,len=30),1),
                    rep(seq(0,20,len=30),1)),nrow=2,byrow=T)

   xpred <- matrix(c(rep(c(1,0),1),rep(c(1,0),1),
                     rep(c(1,1),1),rep(c(1,1),1)),
                     nrow=2,byrow=T)

   colnames(xpred) <- colnames(cbind(xonset,xfailure))

################################################################
# Initial state
################################################################

  state <- NULL

################################################################
# Prior
################################################################

  prior<-list(a0=1,
              b0=1,
              q=0.5,
              mub=10,
              sigmab=200,
              nu=4,
              tinv=diag(1,2),
              nub=6,
              tbinv=diag(1,4),
              m0=rep(0,4),
              S0=diag(100,4),
              maxm=40)
```

```
################################################################
# MCMC
################################################################

  mcmc<-list(nburn=5000,
             nskip=9,
             ndisplay=100,
             nsave=5000,
             tune1=0.25,
             tune2=1)

################################################################
# Fitting the Model
################################################################

  fit1 <- LDPDdoublyint(onset=onset,failure=failure,
                        p=p,xonset=xonset,
                        q=q,xfailure=xfailure,
                        xpred=xpred,grid=grid,
                        prior=prior,
                        mcmc=mcmc,
                        state=state,
                        status=TRUE,
                        work.dir=work.dir)

  fit1
  summary(fit1)

################################################################
# Getting Information for Predictions
################################################################

# Without CI bands and intervals

  fit1.pred <- predict(fit1)
  fit1.pred
  plot(fit1.pred)

# With CI bands and intervals

  fit1.pred <- predict(fit1,compute.band=TRUE)
  fit1.pred
  plot(fit1.pred)

## End(Not run)
```

LDTFPdensity *Density Regression using Linear Dependent Tailfree Processes*

Description

This function generates a posterior density sample for a Linear Dependent Tailfree Process model for conditional density estimation.

Usage

```
LDTFPdensity(y,x,xtf,prediction,prior,mcmc,
             state,status,ngrid=100,
             grid=NULL,compute.band=FALSE,
             type.band="PD",
             data=sys.frame(sys.parent()),
             na.action=na.fail,
             work.dir=NULL)
```

Arguments

`y`	a vector giving the response variables.
`x`	a matrix giving the design matrix for the median function.
`xtf`	a matrix giving the design matrix for the conditional probabilities.
`prediction`	a list giving the information used to obtain conditional inferences. The list includes the following elements: `xdenpred` and `xtfdenpred` giving the design matrices for the median and conditional probabilities, respectively, used to obtain inferences about the conditional densities and survival functions, `xmedpred` and `xtfmedpred` giving the design matrices for the median and conditional probabilities, respectively, used to obtain inferences about quantiles, and `quans` a double precision vector giving THREE quantiles for which inferences are obtained. If `quans` is not specified, the default is quans=c(0.03,0.50,0.97).
`prior`	a list giving the prior information. The list includes the following parameter: `maxn` an integer giving the truncation of the tailfree process, `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the linear dependent tailfree prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing), `mub` giving the mean of the normal prior of the median regression coefficients, `Sb` giving the (co)variance of the normal prior distribution for the median regression coefficents, and `tau1` and `tau2` giving th hyperparameters of the inv-gamma distribution for the centering variance.
`mcmc`	a list giving the MCMC parameters. The list must include the following elements: `nburn` an integer giving the number of burn-in scans, `nskip` an integer giving the thinning interval, `nsave` an integer giving the total number of scans to be saved, `ndisplay` an integer giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.

`ngrid`	integer giving the number of grid points where the conditional density estimate is evaluated. The default is 100.
`grid`	vector of grid points where the conditional densities are evaluated. The default value is NULL and the grid is chosen according to the range of the data.
`compute.band`	logical variable indicating whether the credible band for the conditional density and mean function must be computed.
`type.band`	string indication the type of credible band to be computed; if equal to "HPD" or "PD" then the 95 percent pointwise HPD or PD band is computed, respectively.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `LDTFPdensity` to print an error message and terminate if there are any incomplete observations.
`work.dir`	working directory.

Details

This generic function fits a Linear Dependent Tailfree process (Jara and Hanson, 2011), given by:

$$y_i = x_i'\beta + v_i, i = 1, \ldots, n$$

$$v_i | G_{xtf_i} \sim G_{xtfi}$$

$$\{G_{xtf} : xtf \in \mathcal{X}\} | maxm, \alpha, \sigma^2 \sim LDTFP^{maxm}(h, \Pi^{\sigma^2}, A^{\alpha,\rho})$$

where, h is the logistic CDF, and G_{xtf} is median-zero and centered around an $N(0, \sigma^2)$ distribution. To complete the model specification, independent hyperpriors are assumed,

$$\alpha | a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\sigma^{-2} | \tau_1, \tau_2 \sim Gamma(\tau_1/2, \tau_2/2)$$

The precision parameter, α, of the `LDTFP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

The computational implementation of the model is based on Slice sampling (Neal, 2003).

Value

An object of class `LDTFPdensity` representing the LDTFP model fit. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `beta`, `alpha` and `sigma^2`.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`alpha`	a double precision giving the value of the precision parameter.
`betace`	a vector giving the value of the median regression coefficient.
`sigma^2`	a double precision giving the value of the centering variance.
`betatf`	a matrix giving the regression coefficients for each conditional pribability.

Author(s)

Alejandro Jara <<atjara@uc.cl>>

References

Jara, A., Hanson, T. (2011). A class of mixtures of dependent tail-free processes. Biometrika, 98(3): 553 - 566.

Neal, R. (2003) Slice sampling. Anals of Statistics, 31: 705 - 767.

See Also

DPcdensity, LDDPdensity

Examples

```
## Not run:

    ########################
    # IgG data
    ########################
      data(igg)
      z <- igg$age
      y <- log(igg$igg)

    # Design matrices
      ages1 <- z^2
      ages2 <- 1/ages1

      x <- cbind(rep(1,length(y)),ages1,ages2)
      xtf <- cbind(rep(1,length(y)),ages1,ages2)

      colnames(x) <- c("(Intercept)","age^2","age^{-2}")
      colnames(xtf) <- c("(Intercept)","age^2","age^{-2}")

    # Prediction
      xdpred <- c(11/12,25/12,38/12,52/12,65/12)
      agesp1 <- xdpred^2
      agesp2 <- 1/agesp1
      xdenpred <- cbind(rep(1,length(xdpred)),agesp1,agesp2)
      xtfdenpred <- xdenpred

      xmpred <- seq(0.5,6,len=50)
      agesp1 <- xmpred^2
      agesp2 <- 1/agesp1
      xmedpred <- cbind(rep(1,length(xmpred)),agesp1,agesp2)
      xtfmedpred <- xmedpred

      prediction <- list(xdenpred=xdenpred,
 xtfdenpred=xtfdenpred,
                         xmedpred=xmedpred,
                         xtfmedpred=xtfmedpred,
                         quans=c(0.03,0.50,0.97))

    # Prior information
      Sb <- diag(1000,3)
      mub <- rep(0,3)

      prior <- list(maxm=4,
                    a0=1,
                    b0=1,
                    mub=mub,
                    Sb=Sb,
                    tau1=2.02,
                    tau2=2.02)

    # Initial state
      state <- NULL

    # MCMC parameters

      mcmc <- list(nburn=5000,
                   nsave=5000,
                   nskip=4,
                   ndisplay=200)

    # Fitting the model

      fit1 <- LDTFPdensity(y=y,
   x=x,
   xtf=xtf,
   prediction=prediction,
   prior=prior,
   mcmc=mcmc,
   state=state,
   status=TRUE,
   compute.band=TRUE)

      fit1
      summary(fit1)
      plot(fit1)

    # Plots predictions
    # (conditional densities and quantile functions)

      par(mfrow=c(3,2))

      for(i in 1:5)
      {
         plot(fit1$grid,fit1$densm[i,],lwd=2,
              type="l",lty=1,col="black",xlab="log IgG",ylab="density",
              ylim=c(0,2))
         lines(fit1$grid,fit1$densl[i,],lwd=1,lty=2,col="black")
         lines(fit1$grid,fit1$densu[i,],lwd=1,lty=2,col="black")
      }
```

```
  plot(z,y,ylab="log IgG",xlab="Age (years)")
  lines(xmpred,fit1$qmm[,2],lwd=2)
  lines(xmpred,fit1$qml[,2],lwd=1,lty=2)
  lines(xmpred,fit1$qmu[,2],lwd=1,lty=2)

  lines(xmpred,fit1$qmm[,1],lwd=2)
  lines(xmpred,fit1$qml[,1],lwd=1,lty=2)
  lines(xmpred,fit1$qmu[,1],lwd=1,lty=2)

  lines(xmpred,fit1$qmm[,3],lwd=2)
  lines(xmpred,fit1$qml[,3],lwd=1,lty=2)
  lines(xmpred,fit1$qmu[,3],lwd=1,lty=2)


#######################################
# A simulated data using "perfect"
# simulation (mixture of two normals
# and normal true models).
#######################################

# Functions needed to simulate data
# and to evaluate true models

  findq <- function(true.cdf,target,low,
                    upp,epsilon=0.0000001)
  {
      plow <- true.cdf(low)
      pupp <- true.cdf(upp)
      pcenter <- true.cdf((upp+low)/2)
      err <- abs(pcenter-target)
      i <- 0
      while(err > epsilon)
      {
            i <- i + 1
            if(target< pcenter)
            {
               upp <- (upp+low)/2
               pupp <- pcenter
               pcenter <- true.cdf((upp+low)/2)
               err <- abs(pcenter-target)
            }
            if(target>= pcenter)
            {
               low <- (upp+low)/2
               plow <- pcenter
               pcenter <- true.cdf((upp+low)/2)
               err <- abs(pcenter-target)
            }
        }
        return((upp+low)/2)
  }
  true.dens1 <- function(x)
  {
        0.5*dnorm(x,2.5,sqrt(0.005))+
        0.5*dnorm(x,2.85,sqrt(0.005))
  }

  true.dens2 <- function(x)
  {
        dnorm(x,2.1,sqrt(0.0324))
  }

  true.cdf1 <- function(x)
  {
        0.5*pnorm(x,2.50,sqrt(0.005))+
        0.5*pnorm(x,2.85,sqrt(0.005))
  }

  true.cdf2 <- function(x)
  {
        pnorm(x,2.1,sqrt(0.0324))
  }

# Simulation
  nsim <- 500
  qq <- seq(1,nsim)/(nsim+1)

  y1 <- rep(0,nsim)
  for(i in 1:nsim)
  {
      aa <- findq(true.cdf1,qq[i],low=-6,upp=6)
      y1[i] <- aa
  }

  y2 <- rep(0,nsim)
  for(i in 1:nsim)
  {
      aa <- findq(true.cdf2,qq[i],low=-6,upp=6)
      y2[i] <- aa
  }

  trt <- c(rep(0,nsim),rep(1,nsim))
  y <- c(y1,y2)

# Design matrices
  W1 <- cbind(rep(1,2*nsim),trt)
  W2 <- cbind(rep(1,2*nsim),trt)
  colnames(W1) <- c("(Intercept)","trt")
  colnames(W2) <- c("(Intercept)","trt")

# Design matrix for prediction
  intp <- rep(1,2)
  trtp <- c(0,1)
  zpred <- cbind(intp,trtp)
  prediction <- list(xdenpred=zpred,
                     xtfdenpred=zpred,
                     xmedpred=zpred,
                     xtfmedpred=zpred,
                     quans=c(0.03,0.50,0.97))

# Prior information
  prior <- list(maxm=5,
                a0=1,
                b0=1,
                mub=rep(0,2),
                Sb=diag(1000,2),
                tau1=2.002,
                tau2=2.002)

# Initial state
  state <- NULL

# MCMC parameters
  nburn <- 5000
  nsave <- 5000
  nskip <- 4
  ndisplay <- 200
  mcmc <- list(nburn=nburn,
               nsave=nsave,
               nskip=nskip,
               ndisplay=ndisplay)

# Fitting the model
  fit1 <- LDTFPdensity(y=y,
                       x=W1,
                       xtf=W2,
                       grid=seq(1.2,3.2,len=200),
                       prediction=prediction,
                       prior=prior,
                       mcmc=mcmc,
                       state=state,
                       status=TRUE,
                       compute.band=TRUE)

# Plotting density estimates and true models

  par(cex=1.7,mar=c(4.1, 4.1, 1, 1))
  plot(fit1$grid,fit1$densu[1,],type="l",xlab="y",
       ylab="f(y|x)",lty=2,lwd=3,main="trt=0")
  lines(fit1$grid,fit1$densl[1,],lty=2,lwd=3)
  lines(fit1$grid,fit1$densm[1,],lty=1,lwd=3)
  tmp1 <- true.dens1(fit1$grid)
  lines(fit1$grid,tmp1,lty=1,lwd=3,col="red")

  par(cex=1.7,mar=c(4.1, 4.1, 1, 1))
  plot(fit1$grid,fit1$densu[2,],type="l",xlab="y",
        ylab="f(y|x)",lty=2,lwd=3,main="trt=1")
    lines(fit1$grid,fit1$densl[2,],lty=2,lwd=3)
    lines(fit1$grid,fit1$densm[2,],lty=1,lwd=3)
    tmp1 <- true.dens2(fit1$grid)
    lines(fit1$grid,tmp1,lty=1,lwd=3,col="red")


## End(Not run)
```

LDTFPglmm *Generalized linear mixed model using a linear dependent tailfree prior*

Description

This function generates a posterior density sample for a generalized linear mixed model using a linear dependent tail free prior for the random intercept distribution.

Usage

```
LDTFPglmm(y,x,roffset=NULL,g,family,
          xtf,prediction,prior,mcmc,
          state,status,ngrid=100,
          grid=NULL,compute.band=FALSE,
          type.band="PD",
          data=sys.frame(sys.parent()),
          na.action=na.fail,
          work.dir=NULL)
```

Arguments

y	a vector giving the response variables.
x	a matrix giving the design matrix for the fixed effects. This matrix must include the constant term.
roffset	this can be used to specify an a priori known component to be included in the linear predictor during the fitting (only for poisson and gamma models).
g	a vector giving the group indicator for each observation.
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. The families(links) considered by LDTFPglmm so far are binomial(logit), binomial(probit), Gamma(log), and poisson(log). The gaussian(identity) case is implemented separately in the function LDTFPlmm.
xtf	a matrix giving the design matrix for the conditional probabilities of the random intercepts distributions.

`prediction`	a list giving the information used to obtain conditional inferences. The list includes the following elements: `xpred` and `xtfnpred` giving the design matrices for the median and conditional probabilities, respectively, used to obtain inferences about the conditional densities of the random effects, and `quans` a double precision vector giving THREE quantiles for which inferences are obtained. If `quans` is not specified, the default is quans=c(0.03,0.50,0.97).
`prior`	a list giving the prior information. The list includes the following parameter: `maxm` an integer giving the truncation of the tailfree process, `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the linear dependent tailfree prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing), `mub` giving the mean of the normal prior of the fixed effects, `Sb` giving the (co)variance of the normal prior distribution for the fixed effects, and `taub1` and `taub2` giving th hyperparameters of the inv-gamma distribution for the centering variance.
`mcmc`	a list giving the MCMC parameters. The list must include the following elements: `nburn` an integer giving the number of burn-in scans, `nskip` an integer giving the thinning interval, `nsave` an integer giving the total number of scans to be saved, `ndisplay` an integer giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`ngrid`	integer giving the number of grid points where the conditional density estimate is evaluated. The default is 100.
`grid`	vector of grid points where the conditional densities are evaluated. The default value is NULL and the grid is chosen according to the range of the data.
`compute.band`	logical variable indicating whether the credible band for the conditional density and mean function must be computed.
`type.band`	string indication the type of credible band to be computed; if equal to "HPD" or "PD" then the 95 percent pointwise HPD or PD band is computed, respectively.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `LDTFPdensity` to print an error message and terminate if there are any incomplete observations.
`work.dir`	working directory.

Details

This generic function fits a generalized linear mixed effects model using a linear dependent tailfree prior for the random intercepts (Jara and Hanson, 2011). The linear predictor is modeled as follows:

$$\eta_{ij} = x_{ij}'\beta + b_i, i = 1, \ldots, N, j = 1, \ldots, n_i$$

$$b_i | G_{xtf_i} \sim G_{xtfi}$$

$$\{G_{xtf} : xtf \in \mathcal{X}\} | maxm, \alpha, \sigma_b^2 \sim LDTFP^{maxm}(h, \Pi^{\sigma_b^2}, A^{\alpha,\rho})$$

where, h is the logistic CDF, and G_{xtf} is median-zero and centered around an $N(0, \sigma^2 b)$ distribution. To complete the model specification, independent hyperpriors are assumed,

$$\alpha | a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\sigma_b^{-2} | \tau_{b1}, \tau_{b2} \sim Gamma(\tau_{b1}/2, \tau_{b2}/2)$$

The precision parameter, α, of the `LDTFP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

The full conditional distribution for the fixed effects is updated using a MH step based on an IWLS proposal (see, e.g., Jara, Hanson and Lesaffre, 2009). The remaining parameters are sampled using the slice sampling algorithm (Neal, 2003).

Value

An object of class `LDTFPglmm` representing the LDTFP model fit. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `beta`, `alpha` and `sigma^2_b`.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`alpha`	a double precision giving the value of the precision parameter.
`b`	a double precision giving the value of the random effects.
`beta`	a vector giving the value of the fixed effects.
`sigma2b`	a double precision giving the value of the centering variance.
`betatf`	a matrix giving the regression coefficients for each conditional pribability.

Author(s)

Alejandro Jara <`<atjara@uc.cl>`>

References

Jara, A., Hanson, T. (2011). A class of mixtures of dependent tail-free processes. Biometrika, 98(3): 553 - 566.

Jara, A., Hanson, T., Lesaffre, E. (2009) Robustifying generalized linear mixed models using a new class of mixtures of multivariate Polya trees. Journal of Computational and Graphical Statistics, 18(4): 838-860.

Neal, R. (2003) Slice sampling. Anals of Statistics, 31: 705 - 767.

See Also

LDTFPdensity, LDTFPsurvival

Examples

```
## Not run:
    ###############################################
    # A simulated data using "perfect"
    # simulation from a mixture of two
    # normals and normal true models for
    # the random effects.
    # A Poisson sampling distribution
    # is considered.
    ###############################################

    # Functions needed to simulate random effects
    # and to evaluate true models

      findq <- function(true.cdf,target,low,
                        upp,epsilon=0.0000001)
      {
         plow <- true.cdf(low)
         pupp <- true.cdf(upp)
         pcenter <- true.cdf((upp+low)/2)
         err <- abs(pcenter-target)
         i <- 0
         while(err > epsilon)
         {
               i <- i + 1
               if(target< pcenter)
               {
                  upp <- (upp+low)/2
                  pupp <- pcenter
                  pcenter <- true.cdf((upp+low)/2)
                  err <- abs(pcenter-target)
               }
               if(target>= pcenter)
               {
                  low <- (upp+low)/2
                  plow <- pcenter
                  pcenter <- true.cdf((upp+low)/2)
                  err <- abs(pcenter-target)
               }
           }
           return((upp+low)/2)
      }

      true.dens1 <- function(x)
      {
           0.5*dnorm(x,2.,sqrt(0.005))+
           0.5*dnorm(x,2.85,sqrt(0.005))
      }
```

```
  true.dens2 <- function(x)
  {
       dnorm(x,2.1,sqrt(0.0324))
  }

  true.cdf1 <- function(x)
  {
       0.5*pnorm(x,2.,sqrt(0.005))+
       0.5*pnorm(x,2.85,sqrt(0.005))
  }

  true.cdf2 <- function(x)
  {
       pnorm(x,2.1,sqrt(0.0324))
  }

# Simulation of random effects

  nsubject <- 200
  nsim <- nsubject/2
  qq <- seq(1,nsim)/(nsim+1)
  b1 <- rep(0,nsim)
  for(i in 1:nsim)
  {
      aa <- findq(true.cdf1,qq[i],low=-6,upp=6)
      b1[i] <- aa
  }

  b2 <- rep(0,nsim)
  for(i in 1:nsim)
  {
     aa <- findq(true.cdf2,qq[i],low=-6,upp=6)
     b2[i] <- aa
  }

  trt <- c(rep(0,nsim),rep(1,nsim))
  b <- c(b1,b2)

  xtf <- cbind(rep(1,nsubject),trt)

# Simulation of responses

  ni <- 5
  nrec <- nsubject*ni
  y <- NULL
  g <- NULL
  x <- NULL

  z <- rnorm(nrec)

  ll <- 0
  for(i in 1:nsubject)
```

```
   {
       g <- c(g,rep(i,ni))
       for(j in 1:ni)
       {
           ll <- ll +1
           etaij <- b[i] + 1.2*z[ll]
           ytmp <- rpois(1,exp(etaij))
           y <- c(y,ytmp)
           x <- rbind(x,c(1,trt[i],z[ll]))
       }
  }
  colnames(x) <- c("Intercept","trt","z")

# Design matrix for prediction

  xpred <- rbind(c(1,0,0),c(1,1,0))
  xtfpred <- rbind(c(1,0),c(1,1))

  prediction <- list(xpred=xpred,
                     xtfpred=xtfpred,
                     quans=c(0.03,0.50,0.97))

# Prior information
  prior <- list(maxm=5,
                alpha=0.5,
                mub=rep(0,3),
                Sb=diag(1000,3),
                taub1=2.002,
                taub2=2.002)

# Initial state
  state <- NULL


# MCMC parameters
  nburn <- 4000
  nsave <- 4000
  nskip <- 3
  ndisplay <- 500
  mcmc <- list(nburn=nburn,
               nsave=nsave,
               nskip=nskip,
               ndisplay=ndisplay)


# Fitting the model
  fit1 <- LDTFPglmm(y=y,x=x,g=g,family=poisson(log),
                    xtf=xtf,grid=seq(1.2,3.2,len=200),
                    prediction=prediction,
                    prior=prior,
                    mcmc=mcmc,
                    state=state,
                    status=TRUE,
                        compute.band=TRUE)

    # Plotting density estimates and true models
    # for the random intercepts

      par(cex=1.7,mar=c(4.1, 4.1, 1, 1))
      plot(fit1$grid,fit1$densu[1,],type="l",xlab="y",
           ylab="f(y|x)",lty=2,lwd=3,main="trt=0")
      lines(fit1$grid,fit1$densl[1,],lty=2,lwd=3)
      lines(fit1$grid,fit1$densm[1,],lty=1,lwd=3)
      tmp1 <- true.dens1(fit1$grid)
      lines(fit1$grid,tmp1,lty=1,lwd=3,col="red")

      par(cex=1.7,mar=c(4.1, 4.1, 1, 1))
      plot(fit1$grid,fit1$densu[2,],type="l",xlab="y",
           ylab="f(y|x)",lty=2,lwd=3,main="trt=1")
      lines(fit1$grid,fit1$densl[2,],lty=2,lwd=3)
      lines(fit1$grid,fit1$densm[2,],lty=1,lwd=3)
      tmp1 <- true.dens2(fit1$grid)
      lines(fit1$grid,tmp1,lty=1,lwd=3,col="red")


## End(Not run)
```

`LDTFPsurvival` *Survival Regression using Linear Dependent Tailfree Processes*

Description

This function generates a posterior density sample for a Linear Dependent Tailfree Process model for conditional survival estimation of time-to-event data.

Usage

```
LDTFPsurvival(y,x,xtf,prediction,prior,mcmc,
              state,status,grid=seq(0.01,60,len=100),
              compute.band=FALSE,type.band="PD",
              data=sys.frame(sys.parent()),
              na.action=na.fail,work.dir=NULL)
```

Arguments

`y`	a vector giving the response variables.
`x`	a matrix giving the design matrix for the median function.
`xtf`	a matrix giving the design matrix for the conditional probabilities.
`prediction`	a list giving the information used to obtain conditional inferences. The list includes the following elements: `xdenpred` and `xtfdenpred` giving the design

matrices for the median and conditional probabilities, respectively, used to obtain inferences about the conditional densities and survival functions, `xmedpred` and `xtfmedpred` giving the design matrices for the median and conditional probabilities, respectively, used to obtain inferences about quantiles, and `quans` a double precision vector giving THREE quantiles for which inferences are obtained. If `quans` is not specified, the default is quans=c(0.03,0.50,0.97).

`prior`	a list giving the prior information. The list includes the following parameter: `maxn` an integer giving the truncation of the tailfree process, `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the linear dependent tailfree prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` is missing), `mub` giving the mean of the normal prior of the median regression coefficients, `Sb` giving the (co)variance of the normal prior distribution for the median regression coefficents, and `tau1` and `tau2` giving th hyperparameters of the inv-gamma distribution for the centering variance.
`mcmc`	a list giving the MCMC parameters. The list must include the following elements: `nburn` an integer giving the number of burn-in scans, `nskip` an integer giving the thinning interval, `nsave` an integer giving the total number of scans to be saved, `ndisplay` an integer giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`grid`	vector of grid points where the conditional survival functions are evaluated. The default value is NULL and the grid is chosen according to the range of the data.
`compute.band`	logical variable indicating whether the credible band for the conditional density and mean function must be computed.
`type.band`	string indication the type of credible band to be computed; if equal to "HPD" or "PD" then the 95 percent pointwise HPD or PD band is computed, respectively.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `LDTFPsurvival` to print an error message and terminate if there are any incomplete observations.
`work.dir`	working directory.

Details

This generic function fits a Linear Dependent Tailfree process (Jara and Hanson, 2011), for (potentially) interval-censored data. Let $T_i \in R^+$ be the time-to-event for subject i and set $z_i = \log T_i$. The model for the log time-to.event data is given by:

$$z_i = x_i'\beta + v_i, i = 1, \ldots, n$$

$$v_i | G_{xtf_i} \sim G_{xtfi}$$

$$\{G_{xtf} : xtf \in \mathcal{X}\}|maxm, \alpha, \sigma^2 \sim LDTFP^{maxm}(h, \Pi^{\sigma^2}, A^{\alpha,\rho})$$

where, h is the logistic CDF, and G_{xtf} is median-zero and centered around an $N(0, \sigma^2)$ distribution. To complete the model specification, independent hyperpriors are assumed,

$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\sigma^{-2}|\tau_1, \tau_2 \sim Gamma(\tau_1/2, \tau_2/2)$$

The precision parameter, α, of the `LDTFP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

The computational implementation of the model is based on Slice sampling (Neal, 2003).

Value

An object of class `LDTFPsurvival` representing the LDTFP model fit. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `beta`, `alpha` and `sigma^2`.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`alpha`	a double precision giving the value of the precision parameter.
`betace`	a vector giving the value of the median regression coefficient.
`sigma^2`	a double precision giving the value of the centering variance.
`betatf`	a matrix giving the regression coefficients for each conditional probability.
`z`	a vector giving the current value of the (imputed) survival times.

Author(s)

Alejandro Jara <`<atjara@uc.cl>`>

References

Jara, A., Hanson, T. (2011). A class of mixtures of dependent tail-free processes. Biometrika, 98(3): 553 - 566.

Neal, R. (2003) Slice sampling. Anals of Statistics, 31: 705 - 767.

See Also

`LDTFPdensity`,`LDDPsurvival`

Examples

```
## Not run:

   ############################################################
   # Time to Cosmetic Deterioration of Breast Cancer Patients
   ############################################################

     data(deterioration)
     attach(deterioration)
     y <- cbind(left,right)

   # Design matrix

     x <- cbind(rep(1,length(trt)),trt)
     xtf <- cbind(rep(1,length(trt)),trt)
     colnames(x) <- c("(Intercept)","trt")
     colnames(xtf) <- c("(Intercept)","trt")

   # Prediction

     xdenpred <- cbind(rep(1,2),c(0,1))
     xtfdenpred <- cbind(rep(1,2),c(0,1))
     xmedpred <- cbind(rep(1,2),c(0,1))
     xtfmedpred <- cbind(rep(1,2),c(0,1))

     prediction <- list(xdenpred=xdenpred,
                        xtfdenpred=xtfdenpred,
                        xmedpred=xmedpred,
                        xtfmedpred=xtfmedpred,
                        quans=c(0.03,0.50,0.97))

   # Prior information

     prior <- list(maxm=5,
                   a0=1,
                   b0=1,
                   mub=rep(0,2),
                   Sb=diag(1000,2),
                   tau1=2,002,
                   tau2=2.002)

   # Initial state
     state <- NULL

   # MCMC parameters

     mcmc <- list(nburn=5000,
                  nsave=5000,
                  nskip=4,
                  ndisplay=200)

   # Fitting the model
```

```
    fit1 <- LDTFPsurvival(y=y,
                          x=x,
                          xtf=xtf,
                          prediction=prediction,
                          prior=prior,
                          mcmc=mcmc,
                          state=state,
                          grid=seq(0.01,70,len=200),
                          status=TRUE,
                          compute.band=TRUE)

    fit1
    summary(fit1)
    plot(fit1)

  # Plotting survival functions estimates

    par(cex=1.7,mar=c(4.1, 4.1, 1, 1))
    x1 <- fit1$grid
    y1 <- fit1$survml[1,]
    x2 <- fit1$grid
    y2 <- fit1$survmu[1,]
    aa <- rbind(x2,y2)[, order(-x2, y2)]
    x2 <- aa[1,]
    y2 <- aa[2,]
    plot(fit1$grid,fit1$survmu[1,],type="l",
         xlab="months",ylab="survival",
         lty=1,lwd=2,ylim=c(0,1),col="lightgray")
    polygon(x=c(x1,x2),y=c(y1,y2),border=NA,col="lightgray")
    lines(fit1$grid,fit1$survmm[1,],lty=1,lwd=3)

    par(cex=1.7,mar=c(4.1, 4.1, 1, 1))
    x1 <- fit1$grid
    y1 <- fit1$survml[2,]
    x2 <- fit1$grid
    y2 <- fit1$survmu[2,]
    aa <- rbind(x2,y2)[, order(-x2, y2)]
    x2 <- aa[1,]
    y2 <- aa[2,]
    plot(fit1$grid,fit1$survmu[2,],type="l",
         xlab="months",ylab="survival",
         lty=1,lwd=2,ylim=c(0,1),col="lightgray")
    polygon(x=c(x1,x2),y=c(y1,y2),border=NA,col="lightgray")
    lines(fit1$grid,fit1$survmm[2,],lty=1,lwd=3)

## End(Not run)
```

`nodal` *Nodal Involvement Data*

Description

This data set consider information on the presence of prostatic nodal involvement collected on 53 patients with prostate cancer reported by Brown (1980).

For the sample of prostate cancer patients, a number of possible predictor variables were measured before surgery. The patients then had surgery to determine nodal involvement. It was required to see if nodal involvement could be accurately predicted from the predictor variables and which ones were most important.

Usage

```
data(nodal)
```

Format

A data frame with 53 observations on the following 7 variables.

`id` an ordered factor giving a unique identifier for the subject in the study

`ssln` a numeric vector giving the prostatic nodal involvement that takes the value 1 if cancer had spread to the surrounding lymph nodes and 0 otherwise

`age` a numeric vector giving the age of the patient in years at diagnosis

`acid` a numeric vector giving the level of serum acid phosphate

`xray` a numeric vector giving the result af an X-ray examination, coded 0 if negative and 1 if positive

`size` a numeric vector giving the size of the tumor, coded 0 if small and 1 if large

`grade` a numeric vector giving the pathological grade of the tumor, coded 0 if less serious and 1 if more serious

Source

Brown, B.W. (1980) Prediction analysis for binary data. In Biostatistics Casebook. R.G. Miller, B. Efron, B.W. Brown and L.E. Moses (editors), 3-18. John Wiley.

References

Chib, S. (1995) Marginal Likelihood from the Gibbs output. Journal of the American Statistical Association, 90: 1313 - 1321.

Examples

```
## Not run: 
    # Data
      data(nodal)
      attach(nodal)
      lacid<-log(acid)

    # Initial state
      state <- NULL
    # MCMC parameters
      nburn<-5000
      nsave<-10000
      nskip<-10
      ndisplay<-100
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay,
                   tune=1.1)


    # Prior distribution
      prior <- list(alpha=1,beta0=c(0,rep(0.75,5)),
                    Sbeta0=diag(c(100,rep(25,5)),6))

    # Fit the model
      fit1 <- DPbinary(ssln~age+lacid+xray+size+grade,prior=prior,mcmc=mcmc,
                       state=state,status=TRUE)
      fit1


## End(Not run)
```

`orings` *Challenger Space Shuttle O-Ring Data*

Description

The motivation for collecting this database was the explosion of the USA Space Shuttle Challenger on 28 January, 1986. The Rogers commission concluded that the Challenger accident was caused by gas leak through the 6 o-ring joints of the shuttle. Dalal, Fowlkes and Hoadley (1989) looked at the number of distressed o-rings (among the 6) versus launch temperature (Temperture) and pressure (Pressure) for 23 previous shuttle flights. The previous shuttles were launched at temperatures between 53ºF and 81ºF.

Usage

```
data(orings)
```

Format

A data frame with 138 observations on the following 4 variables.

ThermalDistress a numeric vector indicating wether the o-ring experienced thermal distress

Temperature a numeric vector giving the launch temperature (degrees F)

Pressure a numeric vector giving the leak-check pressure (psi)

Flight a numeric vector giving the temporal order of flight

Source

Dalal, S.R., Fowlkes, E.B., and Hoadley, B. (1989). Risk analysis of space shuttle : Pre-Challenger Prediction of Failure, Journal of the American Statistical Association, 84: 945 - 957.

References

Dalal, S.R., Fowlkes, E.B., and Hoadley, B. (1989). Risk analysis of space shuttle : Pre-Challenger Prediction of Failure, Journal of the American Statistical Association, 84: 945 - 957.

Lavine, M. (1991). Problems in extrapolation illustrated with space shuttle O-ring data. Journal of the American Statistical Association, 86: 919-922.

Martz, H. F., Zimmer, W.J. (1992). The risk of catastrophic failure of the solid rocket boosters on the space shuttle. The American Statistician, 46: 42-47.

Examples

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

Pbinary *Bayesian analysis for a parametric Bernoulli regression model*

Description

This function generates a posterior density sample for a parametric binary regression model.

Usage

```
Pbinary(formula,link="logit",prior,mcmc,state,status,misc=NULL,
         data=sys.frame(sys.parent()),na.action=na.fail)
```

Arguments

formula	a two-sided linear formula object describing the model fit, with the response on the left of a ~ operator and the terms, separated by + operators, on the right.
link	a description of the link function to be used in the model. The links considered by Pbinary so far are *logit* (default), *probit*, *cloglog*, and *cauchy*.
prior	a list giving the prior information. The list includes the following parameters: *beta0* and *Sbeta0* giving the hyperparameters of the normal prior distribution for the regression coefficients.
mcmc	a list giving the MCMC parameters. The list must include the following integers: *nburn* giving the number of burn-in scans, *nskip* giving the thinning interval, *nsave* giving the total number of scans to be saved, *ndisplay* giving the number of saved scans to be displayed on the screen (the function reports on the screen when every *ndisplay* iterations have been carried out), and *tune* giving the Metropolis tuning parameter.

`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object *state*.
`misc`	misclassification information. When used, this list must include two objects, *sens* and *spec*, giving the sensitivity and specificity, respectively. Both can be a vector or a scalar. This information is used to correct for misclassification in the conditional bernoulli model.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `Pbinary` to print an error message and terminate if there are any incomplete observations.

Details

`Pbinary` simulates from the posterior density of a parametric Bernoulli regression model,

$$y_i \sim \mathcal{B}ernoulli(\pi_i)$$

where $\pi_i = F(X_i\beta)$ and F is a distribution function on the real line known as the inverse of the link function in the context of generalized linear models. The links considered by `Pbinary` so far are *logit* (default), *probit*, *cloglog*, and *cauchy*.

To complete the model specification, the following prior distribution is assumed,

$$\beta|\beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

A Metropolis-Hastings step is used to sample the posterior distribution of the regression coefficients. The Metropolis proposal distribution is centered at its current value and the variance-covariance matrix correspond to the variance-covariance matrix of the MLEs times the tunning parameter, *tune*, specified in the *mcmc* list.

When the model considers correction for misclassification, a modified link function is used. The modified link is a function of the sensitivity and specificity of the classification (see, e.g., Jara, Garcia-Zattera and Lesaffre, 2006).

Value

An object of class `Pbinary` representing the parametric regression model fit. Generic functions such as `print`, `plot`, `summary`, `predict`, and `anova` have methods to show the results of the fit. The results include only the regression coefficients, *beta*.

The MCMC samples of the parameters are stored in the object *thetasave*. This object is included in the list *save.state* and is a matrix which can be analyzed directly by functions provided by the coda package.

The list *state* in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set *status=TRUE* and create the list state based on this starting values. In this case the list *state* must include the following objects:

`beta`	giving the value of the regression coefficients.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

References

Jara, A., Garcia-Zattera, M.J., Lesaffre, E. (2006) Semiparametric Bayesian Analysis of Misclassified Binary Data. XXIII International Biometric Conference, July 16-21, Montreal, Canada.

Examples

```
## Not run: 
    # Bioassay Data Example
    # Cox, D.R. and Snell, E.J. (1989). Analysis of Binary Data. 2nd ed.
    # Chapman and Hall. p. 7
    # In this example there are 150 subjects at 5 different stimulus levels,
    # 30 at each level.

      y<-c(rep(0,30-2),rep(1,2),
           rep(0,30-8),rep(1,8),
           rep(0,30-15),rep(1,15),
           rep(0,30-23),rep(1,23),
           rep(0,30-27),rep(1,27))

      x<-c(rep(0,30),
           rep(1,30),
           rep(2,30),
           rep(3,30),
           rep(4,30))

    # Initial state
      state <- NULL

    # MCMC parameters
      nburn<-5000
      nsave<-5000
      nskip<-10
      ndisplay<-1000
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay,
                   tune=1.1)


    # Prior distribution
      prior <- list(beta0=rep(0,2), Sbeta0=diag(10000,2))

    # Fit a logistic regression model
      fit1 <- Pbinary(y~x,link="logit",prior=prior,
                      mcmc=mcmc,state=state,status=TRUE)
      fit1
    # Fit a probit regression model
      fit2 <- Pbinary(y~x,link="probit",prior=prior,
                      mcmc=mcmc,state=state,status=TRUE)
      fit2

    # Fit a cloglog regression model
      fit3 <- Pbinary(y~x,link="cloglog",prior=prior,
                      mcmc=mcmc,state=state,status=TRUE)
      fit3

    # Fit a cauchy regression model
      fit4 <- Pbinary(y~x,link="cauchy",prior=prior,
                      mcmc=mcmc,state=state,status=TRUE)
      fit4

## End(Not run)
```

`Plm` *Bayesian analysis for a parametric linear regression model*

Description

This function generates a posterior density sample from a parametric linear regression model using a normal distribution of the errors.

Usage

```
Plm(formula,prior,mcmc,state,status,
     data=sys.frame(sys.parent()),na.action=na.fail)
```

Arguments

`formula`	a two-sided linear formula object describing the model fit, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right.
`prior`	a list giving the prior information. The list includes the following parameter: `tau1` and `tau2` giving the hyperparameters for the prior distribution of the error variance, `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the regression coefficients.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on the screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.

`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `Plm` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a linear regression model:

$$y_i = X_i\beta + V_i, i = 1, \ldots, n$$

$$V_i|\sigma^2 \sim N(0, \sigma^2)$$

To complete the model specification, independent hyperpriors are assumed,

$$\beta|\beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

$$\sigma^{-2}|\tau_1, \tau_2 \sim Gamma(\tau_1/2, \tau_2/2)$$

Value

An object of class `Plm` representing the parametric linear regression model fit. Generic functions such as `print`, `plot`, `summary`, and `anova` have methods to show the results of the fit. The results include `beta`, and `sigma2`.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`beta`	giving the value of the regression coefficients.
`sigma2`	giving the error variance.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

Examples

```
## Not run:

    ############################################
    # The Australian Institute of Sport's data
    ############################################
      data(sports)
      attach(sports)

    # Initial state
      state <- NULL

    # MCMC parameters

      nburn <- 5000
      nsave <- 10000
      nskip <- 20
      ndisplay <- 100
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,
                   ndisplay=ndisplay)

    # Prior information
      prior <- list(beta0=rep(0,3),
                    Sbeta0=diag(1000,3),
                    tau1=0.01,
                    tau2=0.01)

    # Fit the model

      fit <- Plm(formula=bmi~lbm+gender,prior=prior,mcmc=mcmc,
                  state=state,status=TRUE)

    # Summary with HPD and Credibility intervals
      summary(fit)
      summary(fit,hpd=FALSE)

    # Plot model parameters (to see the plots gradually set ask=TRUE)
      plot(fit)
      plot(fit,nfigr=2,nfigc=2)

    # Table of Pseudo Contour Probabilities
      anova(fit)


## End(Not run)
```

`predict.DPsurvint` *Computes the Survival Curve in a Bayesian analysis for a semiparametric AFT regression model*

Description

This function generates a posterior density sample of the Survival curve from a semiparametric AFT regression model for interval-censored data.

Usage

```
## S3 method for class 'DPsurvint'
predict(object,grid,xnew=NULL,hpd=TRUE, ...)
```

Arguments

`object`	`DPsurvint` fitted model object.
`grid`	a vector of grid points where the survival curve should be evaluated.
`xnew`	an optional matrix containing the value of the covariables with which to predict. If omitted, the baseline survival information is calculated.
`hpd`	a logical variable indicating whether a 95HPD interval is calculated, `TRUE`, or a 95Credibility interval is caculated, `FALSE`, for the survival curve at each grid point. The default value is `TRUE`.
`...`	further arguments to be passed.

Details

This function computes the survival curve based on the fit of a Mixture of Dirichlet process in a AFT regression model for interval censored data (Hanson and Johnson, 2004).

Given a MCMC sample of size J of the parameters, a sample of the predictive survival curve for X is drawn as follows: for the MCMC scan j of the posterior distribution, with $j = 1, \ldots, J$, we sample from:

$$S^{(j)}(t|X, data) \sim Beta(a^{(j)}(t), b^{(j)}(t))$$

where,

$$a^{(j)}(t) = \alpha^{(j)} G_0^{(j)}((t \exp(X\beta^{(j)}), \infty)) + \sum_{i=1}^{n} \delta_{V_i^{(j)}}((t \exp(X\beta^{(j)}), \infty))$$

and

$$b^{(j)}(t) = \alpha^{(j)} + N - a^{(j)}(t)$$

Value

An object of class `predict.DPsurvint` representing the survival information arising from a `DPsurvint` model fit. The results include the posterior mean (`pmean`), the posterior median (`pmedian`), the posterior standard deviation (`psd`), the naive standard error (`pstd`) and the limits of the HPD or credibility intervals, `plinf` and `plsup`.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

References

Doss, H. (1994). Bayesian nonparametric estimation for incomplete data using mixtures of Dirichlet priors. The Annals of Statistics, 22: 1763 - 1786.

Hanson, T., and Johnson, W. (2004) A Bayesian Semiparametric AFT Model for Interval-Censored Data. Journal of Computational and Graphical Statistics, 13: 341-361.

See Also

DPsurvint

Examples

```
## Not run:
    ####################################
    # A simulated Data Set
    ####################################

     ind<-rbinom(100,1,0.5)
     vsim<-ind*rnorm(100,1,0.25)+(1-ind)*rnorm(100,3,0.25)

     x1<-rep(c(0,1),50)
     x2<-rnorm(100,0,1)

     etasim<-x1+-1*x2
     time<-vsim*exp(-etasim)

     y<-matrix(-999,nrow=100,ncol=2)

     for(i in 1:100){
        for(j in 1:15){
         if((j-1)<time[i] & time[i]<=j){
            y[i,1]<-j-1
            y[i,2]<-j
         }
     }
     if(time[i]>15)y[i,1]<-15
     }

    # Initial state
      state <- NULL

    # MCMC parameters

      nburn<-5000
      nsave<-10000
      nskip<-10
      ndisplay<-50
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,
                   ndisplay=ndisplay,tune=0.125)

    # Prior information
      prior <- list(alpha=10,beta0=rep(0,2),Sbeta0=diag(100000,2),
                    m0=0,s0=1,tau1=0.01,tau2=0.01)


    # Fit the model

      fit1 <- DPsurvint(y~x1+x2,prior=prior,mcmc=mcmc,
                        state=state,status=TRUE)
      fit1

    # Summary with HPD and Credibility intervals
      summary(fit1)
      summary(fit1,hpd=FALSE)


    # Plot model parameters
      plot(fit1)
      plot(fit1,nfigr=2,nfigc=2)

    # Plot an specific model parameter
      plot(fit1,ask=FALSE,nfigr=1,nfigc=2,param="x1")
      plot(fit1,ask=FALSE,nfigr=1,nfigc=2,param="mu")


    # Predictive information for baseline survival
      grid<-seq(0.00001,14,0.5)
      pred<-predict(fit1,grid=grid)

    # Plot Baseline information with and without Credibility band
      plot(pred)
      plot(pred,band=TRUE)

    # Predictive information with covariates
      npred<-10
      xnew<-cbind(rep(1,npred),seq(-1.5,1.5,length=npred))
      xnew<-rbind(xnew,cbind(rep(0,npred),seq(-1.5,1.5,length=npred)))
      grid<-seq(0.00001,14,0.5)
      pred<-predict(fit1,xnew=xnew,grid=grid)

    # Plot Baseline information
      plot(pred,band=TRUE)

## End(Not run)
```

predict.HDPMcdensity *Predictive Information for the Dependent Random Probability Measures.*

Description

Plot the probability measures arising from a HDPM of normals model for conditional density estimation. Support provided by the NIH/NCI R01CA75981 grant.

Usage

```
## S3 method for class 'HDPMcdensity'
predict(object,pred,i,r,ask=TRUE,nfigr=2,nfigc=2, ...)
```

Arguments

`object`	`HDPMcdensity` fitted model object.
`pred`	indicator for the values of the predictors, given by the row pred in xpred, for which the conditional densities must be drawn.
`i`	study indicator.
`r`	indicator for including (0) or not (1) the common measure.
`ask`	logical variable indicating whether the plots must be displayed sequentially or not.
`nfigr`	number of rows in the figure.
`nfigc`	number of columns in the figure.
`...`	further arguments to be passed.

Details

Must run `HDPMcdensity` first to generate posterior simulations.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

Peter Mueller <<`pmueller@mdanderson.org`>>

References

Mueller, P., Quintana, F. and Rosner, G. (2004). A Method for Combining Inference over Related Nonparametric Bayesian Models. Journal of the Royal Statistical Society, Series B, 66: 735-749.

See Also

`HDPMcdensity`

Examples

```
## Not run:
    # Data
      data(calgb)
      attach(calgb)
      y <- cbind(Z1,Z2,Z3,T1,T2,B0,B1)
      x <- cbind(CTX,GM,AMOF)

      z <- cbind(y,x)

    #  Data for prediction
      data(calgb.pred)
      xpred <- as.matrix(calgb.pred[,8:10])


    # Prior information
      prior <- list(pe1=0.1,
                    pe0=0.1,
                    ae=1,
                    be=1,
                    a0=rep(1,3),
                    b0=rep(1,3),
                    nu=12,
                    tinv=0.25*var(z),
    m0=apply(z,2,mean),
                    S0=var(z),
    nub=12,
                    tbinv=var(z))


    # Initial state
      state <- NULL

    # MCMC parameters

      mcmc <- list(nburn=5000,
                   nsave=5000,
                   nskip=3,
                   ndisplay=100)

    # Fitting the model
      fit1 <- HDPMcdensity(formula=y~x,
                          study=~study,
                          xpred=xpred,
                          prior=prior,
                          mcmc=mcmc,
                          state=state,
                          status=TRUE)

    # Posterior inference
      fit1
      summary(fit1)

    # Plot the parameters
    # (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE)

    # Plot the a specific parameters
    # (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE,param="eps",nfigr=1,nfigc=2)

    # Plot the measure for each study
    # under first values for the predictors, xpred[1,]
      predict(fit1,pred=1,i=1,r=1) # pred1, study 1
      predict(fit1,pred=1,i=2,r=1) # pred1, study 2

    # Plot the measure for each study
    # under second values for the predictors, xpred[2,]
      predict(fit1,pred=2,i=1,r=1) # pred2, study 1
      predict(fit1,pred=2,i=2,r=1) # pred2, study 2
```

```
    # Plot the idiosyncratic measure for each study
    # under first values for the predictors, xpred[1,]
      predict(fit1,pred=1,i=1,r=0) # study 1
      predict(fit1,pred=1,i=2,r=0) # study 2

    # Plot the common measure
    # under first values for the predictors, xpred[1,]
      predict(fit1,pred=1,i=0)

## End(Not run)
```

`predict.HDPMdensity` *Predictive Information for the Dependent Random Probability Measures.*

Description

Plot the probability measures arising from a HDPM of normals model. Support provided by the NIH/NCI R01CA75981 grant.

Usage

```
## S3 method for class 'HDPMdensity'
predict(object,i,r,ask=TRUE,nfigr=2,nfigc=2, ...)
```

Arguments

`object`	`HDPMdensity` fitted model object.
`i`	study indicator.
`r`	indicator for including (0) or not (1) the common measure.
`ask`	logical variable indicating whether the plots must be displayed sequentially or not.
`nfigr`	number of rows in the figure.
`nfigc`	number of columns in the figure.
`...`	further arguments to be passed.

Details

Must run `HDPMdensity` first to generate posterior simulations.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

Peter Mueller <<`pmueller@mdanderson.org`>>

References

Mueller, P., Quintana, F. and Rosner, G. (2004). A Method for Combining Inference over Related Nonparametric Bayesian Models. Journal of the Royal Statistical Society, Series B, 66: 735-749.

See Also

`HDPMdensity`

Examples

```
## Not run:

    # Data
      data(calgb)
      attach(calgb)
      y <- cbind(Z1,Z2,Z3,T1,T2,B0,B1)

    # Prior information
      prior <- list(pe1=0.1,
                    pe0=0.1,
                    ae=1,
                    be=1,
                    a0=rep(1,3),
                    b0=rep(1,3),
                    nu=9,
                    tinv=0.25*var(y),
      m0=apply(y,2,mean),
                    S0=var(y),
      nub=9,
                    tbinv=var(y))


    # Initial state
      state <- NULL

    # MCMC parameters

      mcmc <- list(nburn=5000,
                   nsave=5000,
                   nskip=3,
                   ndisplay=100)

    # Fitting the model
      fit1 <- HDPMdensity(y=y,
                          study=study,
                          prior=prior,
                          mcmc=mcmc,
                          state=state,
                          status=TRUE)

    # Posterior inference
      fit1
      summary(fi1)

    # Plot the parameters
    # (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE)

    # Plot the a specific parameters
    # (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE,param="eps",nfigr=1,nfigc=2)

   # Plot the measure for each study
      predict(fit1,i=1,r=1) # study 1
      predict(fit1,i=2,r=1) # study 2

    # Plot the idiosyncratic measure for each study
      predict(fit1,i=1,r=0) # study 1
      predict(fit1,i=2,r=0) # study 2

    # Plot the common measure
      predict(fit1,i=0)

## End(Not run)
```

`ps` *Specify a smoothing spline fit in a PSgam formula*

Description

A symbolic wrapper to indicate a smooth term in a formula argument to PSgam

Usage

```
ps(..., k=50,degree=3,pord=1)
```

Arguments

`...`	the predictors.
`k`	an integer giving the number of knots. The number of basis functions is k+degree.
`degree`	an integer giving the degree of the B-splines, e.g. degree=3 gives a cubic spline.
`pord`	an integer giving the order of difference penalty.

Value

ps returns the vectors of predictors, endowed with a number of attributes. The vector itself is used in the construction of the model matrix.

Note that ps does not do the smoothing; it simply sets things up for `PSgam`.

Author(s)

Alejandro Jara <<atjara@uc.cl>>

References

Eilers, P.H.C. and Marx, B.D. (1996) Flexible Smoothing with B-splines and penalties. Statistical Science, 11(2): 89-121.

Examples

```
    # fit Start using a smoothing spline with 4 df.
      y ~ Age + ps(Start, degree=4)
    # fit log(Start) using a smoothing spline with 5 df.
      y ~ Age + ps(log(Start), degree=5)
```

`PSgam` *Bayesian analysis for a semiparametric generalized additive model*

Description

This function generates a posterior density sample for a semiparametric generalized additive model, using a B-Splines and penalties.

Usage

```
PSgam(formula,family=gaussian(),offset,n,prior,mcmc,state,status,
       ngrid=20,data=sys.frame(sys.parent()),na.action=na.fail)
```

Arguments

`formula`	a two-sided linear formula object describing the linear predictor of the model, with the response on the left of a ~ operator and the terms, separated by + operators, on the right. Built-in nonparametric smoothing terms are indicated by `ps` for smoothing splines terms. See the documentation for `ps` for its arguments.
`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. The families(links) considered by `PSgam` so far are gaussian(identity), binomial(logit), binomial(probit), Gamma(log), and poisson(log).
`offset`	this can be used to specify an a priori known component to be included in the linear predictor during the fitting (only for poisson and gamma models).
`n`	this can be used to indicate the total number of cases in a binomial model (only implemented for the logistic link). If it is not specified the response variable must be binary.

`prior`	a list giving the prior information. The list include the following parameters: `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the parametric part of the model, `taub1` and `taub2` giving the hyperparameters for the prior distribution of the inverse of the penalty parameters (the same for all), and `tau1` and `tau2` giving the hyperparameters for the prior distribution of the inverse of the dispersion parameter (only gaussian and Gamma models).
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`ngrid`	number of grid points where the smoothers are evaluated. The default value is 20.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `PSgam` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a generalized additive model (see, e.g., Hastie and Tibshirani, 1990) using Penalized splines (see, e.g., Eilers and Marx, 1996; Lang and Brezger, 2004). The linear predictor is modeled as follows:

$$\eta_i = X_i\beta + f_1(x_{1i}) + ... + f_p(x_{pi}), i = 1, \ldots, n$$

where the effect f of the a covariate x is approximated by a polinomial spline with equally spaced knots, written in terms of a linear combination of B-spline basis functions. Specifically, the function f is aproximated by a spline of degree l with r equally spaced knots within the domain of x. It is well known that this spline can be written in terms of a linear combination of $q = l + r$ B-spline basis,

$$f(x) = \sum_{j=1}^{q} b_j B_j(x).$$

The computational implementation of the model is model-specific.

For the `poisson`, `Gamma`, and `binomial(logit)`, the full conditional distributions for fixed and random effects are generated through the Metropolis-Hastings algorithm with a IWLS proposal (see, West, 1985 and Gamerman, 1997).

For the `binomial(probit)` case the following latent variable representation is used:

$$y_{ij} = I(w_{ij} > 0), j = 1, \ldots, n_i.$$

Value

An object of class `PSgam` representing the generalized additive model fit. Generic functions such as `anova`, `print`, `plot`, and `summary` have methods to show the results of the fit. The results include the parametric component of the linear predictor `beta`, the dispersion parameter of the Gamma or gaussian model, and the penalty parameters `sigmab`.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`b`	a vector of dimension q giving the value of the B-spline coefficients.
`beta`	giving the value of the parametric components of the linear predictor.
`sigmab`	giving the penalty parameters.
`phi`	giving the dispersion parameter for the Gamma or gaussian model (if needed).

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

References

Eilers, P.H.C. and Marx, B.D. (1996) Flexible Smoothing with B-splines and penalties. Statistical Science, 11(2): 89-121.

Gamerman, D. (1997) Sampling from the posterior distribution in generalized linear mixed models. Statistics and Computing, 7: 57-68.

Hastie, T. and Tibshirani, R. (1990) Generalized Additive Models. London: Chapman and Hall.

Lang, S., Brezger, A. (2004) Bayesian P-Splines Journal of Computational and Graphical Statistics, 13: 183-212.

West, M. (1985) Generalized linear models: outlier accomodation, scale parameter and prior distributions. In Bayesian Statistics 2 (eds Bernardo et al.), 531-558, Amsterdam: North Holland.

Examples

```
## Not run:

 # Normal simulated data
   set.seed(0)
   n <- 400
   sig <- 2
   x0 <- runif(n, 0, 1)
   x1 <- runif(n, 0, 1)
   x2 <- runif(n, 0, 1)
   x3 <- runif(n, 0, 1)
   f0 <- function(x) 2 * sin(pi * x)
   f1 <- function(x) exp(2 * x)
   f2 <- function(x) 0.2*x^11*(10*(1-x))^6+10*(10*x)^3*(1-x)^10
   f3 <- function(x) 0*x
```

```
  f <- f0(x0) + f1(x1) + f2(x2)
  e <- rnorm(n, 0, sig)
  y <- f + e

# prior
  prior <- list(taub1=2.02,
               taub2=0.02,
               beta0=rep(0,1),
               Sbeta0=diag(100,1),
               tau1=6.01,
               tau2=2.01)

 # Initial state
   state <- NULL

 # MCMC parameters
   nburn <- 5000
   nsave <- 5000
   nskip <- 0
   ndisplay <- 100
   mcmc <- list(nburn=nburn,
               nsave=nsave,
               nskip=nskip,
               ndisplay=ndisplay)


 # fitting the model
   fit1 <- PSgam(formula=y~ps(x0,x1,x2,x3,k=20,degree=3,pord=1),
                family=gaussian(),prior=prior,
                mcmc=mcmc,ngrid=30,
                state=state,status=TRUE)


 # A binary example
   g <- (f-5)/3
   g <- binomial()$linkinv(g)
   y <- rbinom(n,1,g)

 # fitting the model
   fit2 <- PSgam(formula=y~ps(x0,x1,x2,x3,k=20,degree=3,pord=1),
                family=binomial(logit),prior=prior,
                mcmc=mcmc,ngrid=30,
                state=state,status=TRUE)

 # Poisson data
   g <- exp(f/4)
   y <- rpois(n,g)

 # fitting the model
   fit3 <- PSgam(formula=y~ps(x0,x1,x2,x3,k=20,degree=3,pord=1),
                family=poisson(log),prior=prior,
                mcmc=mcmc,ngrid=30,
                state=state,status=TRUE)
```

```
## End(Not run)
```

`psychiatric` *Psychiatric Clinical Trial*

Description

This data set consider information from a pychiatric clinical trial, reported by Hedeker and Gibbons (1994) and collected in the NIMH Schizophrenia Collaborative Study on treatment-related changes in overall severity. For each patient, the Impatient Multidimensional Psychiatric Scale was scored. In this study, 437 patients were randomly assigned to receive one of four medications: placebo, chlorpromazine, fluphenazine, or thioridazine.

Usage

```
data(psychiatric)
```

Format

A data frame with 1603 observations on the following 5 variables.

`id` a numeric vector giving a unique identifier for the subject in the study

`imps79o` a factor giving the Impatient Multidimensional Psuchiatric Scale scored as `1` = normal or borderline mentally ill, `2` = midly or moderately ill, `3` = markadely ill, and `4` = severely or among the the most extremely ill

`tx` a numeric vector giving the treatment group, 0=Placebo, 1=Drug

`week` a numeric vector giving the week where the measurement ocurred

`sweek` a numeric vector giving the square root transformation of time

Source

Hedeker, D. and Gibbons, R.D. (1994) A random-effects ordinal regression model for multilevel data. Biometrics, 50: 933-944.

References

Hedeker, D. and Gibbons, R.D. (1994) A random-effects ordinal regression model for multilevel data. Biometrics, 50: 933-944.

Examples

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

PTdensity	*Nonparametric Bayesian density estimation using Mixtures of Polya Trees*

Description

This function generates a posterior density sample for a Mixture of Polya trees model.

Usage

```
PTdensity(y,ngrid=1000,grid=NULL,prior,mcmc,state,status,
          data=sys.frame(sys.parent()),na.action=na.fail)
```

Arguments

y	a vector or matrix giving the data from which the density estimate is to be computed.
ngrid	number of grid points where the density estimate is evaluated. This is only used if dimension of y is lower or equal than 2. The default value is 1000.
grid	matrix of dimension ngrid*nvar of grid points where the density estimate is evaluated. This is only used if dimension of y is lower or equal than 2. The default value is NULL and the grid is chosen according to the range of the data.
prior	a list giving the prior information. The list includes the following parameter: a0 and b0 giving the hyperparameters for prior distribution of the precision parameter of the Poly tree prior, alpha giving the value of the precision parameter (it must be specified if alpha is missing, see details below), optionally M giving the finite level to be considered (if M is specified, a partially specified mixture of Polya trees model is fitted), nu0 and tinv or tau1 and tau2 giving the hyperparameters of the inverted Wishart or inverted gamma prior distribution for the centering covariance or variance, respectively, sigma giving the value of the standard deviation (univariate case) or covariance matrix (multivariate case) of the centering distribution (if missing and if nu0 and tinv or tau1 and tau2 are missing, Jeffrey's prior is used for the centering (co)variance matrix, m0 and S0 giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, and mu giving the value of the mean of the centering distribution (if missing and if m0 and S0 are missing, Jeffery's prior is used for mu).
mcmc	a list giving the MCMC parameters. The list must include the following integers: nburn giving the number of burn-in scans, nskip giving the thinning interval, nsave giving the total number of scans to be saved, ndisplay giving the number of saved scans to be displayed on screen (the function reports on the screen when every ndisplay iterations have been carried out), tune1, tune2, and tune3 giving the positive Metropolis tuning parameter for the baseline mean, variance, and precision parameter, respectively (the default value is 1.1)

`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `PTdensity` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a Mixture of Polya Trees prior for the density estimation (see, e.g., Lavine, 1992 and 1994; Hanson, 2006). In the univariate case, the model is given by:

$$Y_1, \ldots, Y_n | G \sim G$$

$$G | \alpha, \mu, \sigma \sim PT(\Pi^{\mu,\sigma^2}, A)$$

where, the the PT is centered around a $N(\mu, \sigma^2)$ distribution, by taking each m level of the partition Π^{μ,σ^2} to coincide with the $k/2^m, k = 0, \ldots, 2^m$ quantile of the $N(\mu, \sigma^2)$ distribution. The family $A = \{\alpha_e : e \in E^*\}$, where $E^* = \bigcup_{m=0}^{m} E^m$ and E^m is the m-fold product of $E = \{0, 1\}$, was specified as $\alpha_{e_1 \ldots e_m} = \alpha m^2$.

Analogous to the univariate model, in the multivariate case the PT prior is characterized by partitions of R^d, and a collection of conditional probabilities that link sets in adjacent tree levels, i.e., they link each parent set in a given level to its 2^d offpring stes in the subsequent level. The multivariate model is given by:

$$Y_1, \ldots, Y_n | G \sim G$$

$$G | \alpha, \mu, \Sigma \sim PT(\Pi^{\mu,\Sigma}, A)$$

where, the the PT is centered around a $N_d(\mu, \Sigma)$ distribution. In this case, the class of partitions that we consider, starts with base sets that are Cartesian products of intervals obtained as quantiles from the standard normal distribution. A multivariate location-scale transformation, $Y = \mu + \Sigma^{1/2} z$, is applied to each base set yielding the final sets.

A Jeffry's prior can be specified for the centering parameters,

$$f(\mu, \sigma^2 \propto \sigma^{-2}$$

and

$$f(\mu, \Sigma) \propto |\Sigma|^{-(d+1)/2}$$

in the univariate and multivariate case, respectively. Alternatively, the centering parameters can be fixed to user-specified values or proper priors can be assigned. In the univariate case, the following proper priors can be assigned:

$$\mu | m_0, S_0 \sim N(m_0, S_0)$$

$$\sigma^{-2} | \tau_1, \tau_2 \sim \Gamma(\tau_1/2, \tau_2/2)$$

In the multivariate case, the following proper priors can be assigned:

$$\mu | m_0, S_0 \sim N(m_0, S_0)$$

$$\Sigma|\nu_0, T \sim IW(\nu_0, T)$$

Note that the inverted-Wishart prior is parametrized such that $E(\Sigma) = T^{-1}/(\nu_0 - q - 1)$.

To complete the model specification, independent hyperpriors are assumed,

$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$

The precision parameter, α, of the `PT` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

In the computational implementation of the model, Metropolis-Hastings steps are used to sample the posterior distribution of the baseline and precision parameters.

Value

An object of class `PTdensity` representing the Polya tree model fit. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `mu`, `sigma` or `Sigma` in the univariate or multivariate case, respectively, and the precision parameter `alpha`.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`mu`	giving the value of the baseline mean.
`sigma`	giving the baseline standard deviation or the baseline covariance matrix in the univariate or multivariate case, respectively.
`alpha`	giving the value of the precision parameter.

Author(s)

Alejandro Jara <`<atjara@uc.cl>`>

Tim Hanson <`<hansont@stat.sc.edu>`>

References

Hanson, T. (2006) Inference for Mixtures of Finite Polya Trees. Journal of the American Statistical Association, 101: 1548-1565.

Lavine, M. (1992) Some aspects of Polya tree distributions for statistical modelling. The Annals of Statistics, 20: 1222-11235.

Lavine, M. (1994) More aspects of Polya tree distributions for statistical modelling. The Annals of Statistics, 22: 1161-1176.

See Also

`DPdensity`, `BDPdensity`

Examples

```
## Not run:
    ####################################
    # Univariate example
    ####################################

    # Data
      data(galaxy)
      galaxy<-data.frame(galaxy,speeds=galaxy$speed/1000)
      attach(galaxy)

    # Initial state
      state <- NULL

    # MCMC parameters
      nburn <- 2000
      nsave <- 5000
      nskip <- 49
      ndisplay <- 500
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay,
                   tune1=0.03,tune2=0.25,tune3=1.8)

    # Prior information
      prior<-list(a0=1,b0=0.01,M=6,m0=21,S0=100,sigma=20)

    # Fitting the model

      fit1 <- PTdensity(y=speeds,ngrid=1000,prior=prior,mcmc=mcmc,
                        state=state,status=TRUE)

    # Posterior means
      fit1

    # Plot the estimated density
      plot(fit1,ask=FALSE)
      points(speeds,rep(0,length(speeds)))

    # Plot the parameters
    # (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE,output="param")

    # Extracting the density estimate
      cbind(fit1$x1,fit1$dens)


    ####################################
    # Bivariate example
    ####################################

    # Data
      data(airquality)
      attach(airquality)
      ozone <- Ozone**(1/3)
      radiation <- Solar.R

    # Prior information
      prior <- list(a0=5,b0=1,M=4,
                    m0=c(0,0),S0=diag(10000,2),
                    nu0=4,tinv=diag(1,2))

    # Initial state
      state <- NULL

    # MCMC parameters
      nburn <- 2000
      nsave <- 5000
      nskip <- 49
      ndisplay <- 500
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay,
                   tune1=0.8,tune2=1.0,tune3=1)

    # Fitting the model
      fit1 <- PTdensity(y=cbind(radiation,ozone),prior=prior,mcmc=mcmc,
                        state=state,status=TRUE,na.action=na.omit)

      fit1

    # Plot the estimated density
      plot(fit1)

    # Extracting the density estimate
      x1 <- fit1$x1
      x2 <- fit1$x2
      z <- fit1$dens
      par(mfrow=c(1,1))
      contour(x1,x2,z)
      points(fit1$y)


## End(Not run)
```

`PTglmm` *Bayesian analysis for a semiparametric generalized linear mixed model using a MMPT*

Description

This function generates a posterior density sample for a semiparametric generalized linear mixed model, using a Mixture of Multivariate Polya Trees prior for the distribution of the random effects.

Usage

```
PTglmm(fixed,random,family,offset,n,prior,mcmc,state,status,
      data=sys.frame(sys.parent()),na.action=na.fail)
```

Arguments

`fixed`	a two-sided linear formula object describing the fixed-effects part of the model, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right.
`random`	a one-sided formula of the form `~z1+...+zn | g`, with `z1+...+zn` specifying the model for the random effects and `g` the grouping variable. The random effects formula will be repeated for all levels of grouping.
`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. The families(links) considered by `PTglmm` so far are binomial(logit), binomial(probit), Gamma(log), and poisson(log). The gaussian(identity) case is implemented separately in the function `PTlmm`.
`offset`	this can be used to specify an a priori known component to be included in the linear predictor during the fitting (only for poisson and gamma models).
`n`	this can be used to indicate the total number of cases in a binomial model (only implemented for the logistic link). If it is not specified the response variable must be binary.
`prior`	a list giving the prior information. The list include the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Polya Tree (PT) prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` and `b0` are missing, see details below), `nu0` and `tinv` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix of the normal baseline distribution, `sigma` giving the value of the covariance matrix of the centering distribution (it must be specified if `nu0` and `tinv` are missing), `mub` and `Sb` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `mu` giving the value of the mean of the centering distribution (it must be specified if `mub` and `Sb` are missing), `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the fixed effects (must be specified only if fixed effects are considered in the model), `M` giving the finite level of the PT prior to be considered, and `frstlprob` a logical variable indicating whether the first level probabilities of the PT are fixed or not (the default is FALSE), `tau1` and `tau2` giving the hyperparameters for the prior distribution for the inverse of the precision parameter of the Gamma model (they must be specified only if the Gamma model is considered), and `typep` indicating whether the type of decomposition of the centering covariance matrix is random (1) or not (0).
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on

	the screen when every `ndisplay` iterations have been carried out), `nbase` giving the number scans to be performed before the parameters of the centering distribution and the precision parameter are updated (i.e., the update of this parameters is invoked only once in every `nbase` scans) (the default value is 1), `tune1`, `tune2`, `tune3`, `tune4` and `tune5` giving the Metropolis tuning parameter for the baseline mean, variance, precision parameter, partition and dispersion parameter (only for the Gamma mode), respectively. If `tune1`, `tune2`, `tune3` or `tune4` are not specified or negative, an adpative Metropolis algorithm is performed. If `tune5` is not specified, a default value of 1.1 is assumed. Finally, the integer `samplef` indicates whether the functional parameters must be sample (1) or not (0).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `PTglmm` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a generalized linear mixed-effects model using a Mixture of Multivariate Polya Trees prior (see, Lavine 1992; 1994, for details about univariate PT) for the distribution of the random effects as described in Jara, Hanson and Lesaffre (2009). The linear predictor is modeled as follows:

$$\eta_i = X_i\beta_F + Z_i\beta_R + Z_i b_i, i = 1, \ldots, n$$

$$\theta_i|G \sim G$$

$$G|\alpha, \mu, \Sigma, O \sim PT^M(\Pi^{\mu,\Sigma,O}, \mathcal{A})$$

where, $\theta_i = \beta_R + b_i$, $\beta = \beta_F$, and O is an orthogonal matrix defining the decomposition of the centering covariance matrix. As in Hanson (2006), the PT prior is centered around the $N_d(\mu, \Sigma)$ distribution. However, we consider the class of partitions $\Pi^{\mu,\Sigma,O}$. The partitions starts with base sets that are Cartesian products of intervals obtained as quantiles from the standard normal distribution. A multivariate location-scale transformation $\theta = \mu + \Sigma^{1/2}z$ is applied to each base set yielding the final sets. Here $\Sigma^{1/2} = T'O'$, where T is the unique upper triangular Cholesky matrix of Σ. The family $\mathcal{A} = \{\alpha_e : e \in E^*\}$, where $E^* = \bigcup_{m=0}^{M} E_d^m$, with E_d and E_d^m the d-fold product of $E = \{0, 1\}$ and the the m-fold product of E_d, respectively. The family $\mathcal{A}$ was specified as $\alpha_{e_1 \ldots e_m} = \alpha m^2$.

To complete the model specification, independent hyperpriors are assumed,

$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\beta|\beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

$$\mu|\mu_b, S_b \sim N(\mu_b, S_b)$$

$$\Sigma|\nu_0, T \sim IW(\nu_0, T)$$
$$O \sim Haar(q)$$

Note that the inverted-Wishart prior is parametrized such that $E(\Sigma) = T^{-1}/(\nu_0 - q - 1)$.

The precision parameter, α, of the PT prior can be considered as random, having a gamma distribution, $Gamma(a_0, b_0)$, or fixed at some particular value.

The inverse of the dispersion parameter of the Gamma model is modeled using gamma distribution, $\Gamma(\tau_1/2, \tau_2/2)$.

The computational implementation of the model is based on the marginalization of the PT as discussed in Jara, Hanson and Lesaffre (2009).

Value

An object of class PTglmm representing the generalized linear mixed-effects model fit. Generic functions such as print, plot, and summary have methods to show the results of the fit. The results include betaR, betaF, mu, the elements of Sigma, the precision parameter alpha, the dispersion parameter of the Gamma model, and ortho.

The function PTrandom can be used to extract the posterior mean of the random effects.

The list state in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set status=TRUE and create the list state based on this starting values. In this case the list state must include the following objects:

alpha	giving the value of the precision parameter.
b	a matrix of dimension (nsubjects)*(nrandom effects) giving the value of the random effects for each subject.
beta	giving the value of the fixed effects.
mu	giving the mean of the normal baseline distributions.
sigma	giving the variance matrix of the normal baseline distributions.
phi	giving the precision parameter for the Gamma model (if needed).
ortho	giving the orthogonal matrix H, used in the decomposition of the covariance matrix.

Author(s)

Alejandro Jara <<atjara@uc.cl>>

Tim Hanson <<hansont@stat.sc.edu>>

References

Hanson, T. (2006) Inference for Mixtures of Finite Polya Trees. Journal of the American Statistical Association, 101: 1548-1565.

Jara, A., Hanson, T., Lesaffre, E. (2009) Robustifying Generalized Linear Mixed Models using a New Class of Mixtures of Multivariate Polya Trees. Journal of Computational and Graphical Statistics, 18(4): 838-860.

Lavine, M. (1992) Some aspects of Polya tree distributions for statistical modelling. The Annals of Statistics, 20: 1222-11235.

Lavine, M. (1994) More aspects of Polya tree distributions for statistical modelling. The Annals of Statistics, 22: 1161-1176.

See Also

PTrandom, PTglmm , PTolmm, DPMglmm, DPMlmm, DPMolmm, DPlmm , DPglmm, DPolmm

Examples

```
## Not run:
    # Respiratory Data Example
      data(indon)
      attach(indon)

      baseage2 <- baseage**2
      follow <- age-baseage
      follow2 <- follow**2

    # Prior information

      prior <- list(alpha=1,
                    M=4,
                    frstlprob=FALSE,
                    nu0=4,
                    tinv=diag(1,1),
                    mub=rep(0,1),
                    Sb=diag(1000,1),
                    beta0=rep(0,9),
                    Sbeta0=diag(10000,9))

    # Initial state
      state <- NULL

    # MCMC parameters

      nburn <- 5000
      nsave <- 5000
      nskip <- 20
      ndisplay <- 100
      mcmc <- list(nburn=nburn,
                   nsave=nsave,
                   nskip=nskip,
                   ndisplay=ndisplay,
                   tune1=0.5,tune2=0.5,
                   samplef=1)

    # Fitting the Logit model
      fit1 <- PTglmm(fixed=infect~gender+height+cosv+sinv+xero+baseage+baseage2+
                     follow+follow2,random=~1|id,family=binomial(logit),
                     prior=prior,mcmc=mcmc,state=state,status=TRUE)
      fit1

      plot(PTrandom(fit1,predictive=TRUE))

    # Plot model parameters (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE)
      plot(fit1,ask=FALSE,nfigr=2,nfigc=2)

    # Extract random effects
      PTrandom(fit1)
      PTrandom(fit1,centered=TRUE)

    # Extract predictive information of random effects
      PTrandom(fit1,predictive=TRUE)

    # Predictive marginal and joint distributions
      plot(PTrandom(fit1,predictive=TRUE))

    # Fitting the Probit model
      fit2 <- PTglmm(fixed=infect~gender+height+cosv+sinv+xero+baseage+baseage2+
                     follow+follow2,random=~1|id,family=binomial(probit),
                     prior=prior,mcmc=mcmc,state=state,status=TRUE)
      fit2

    # Plot model parameters (to see the plots gradually set ask=TRUE)
      plot(fit2,ask=FALSE)
      plot(fit2,ask=FALSE,nfigr=2,nfigc=2)

    # Extract random effects
      PTrandom(fit2)

    # Extract predictive information of random effects
      PTrandom(fit2,predictive=TRUE)

    # Predictive marginal and joint distributions
      plot(PTrandom(fit2,predictive=TRUE))

## End(Not run)
```

`PTlm` *Bayesian analysis for a semiparametric linear regression model*

Description

This function generates a posterior density sample from a semiparametric linear regression model using a Mixture of Polya Trees prior for the distribution of the errors.

Usage

```
PTlm(formula,ngrid=200,grid=NULL,prior,mcmc,state,status,
     data=sys.frame(sys.parent()),na.action=na.fail)
```

Arguments

`formula`	a two-sided linear formula object describing the model fit, with the response on the left of a ~ operator and the terms, separated by + operators, on the right.
`ngrid`	number of grid points where the error density estimate is evaluated. The default value is 200.
`grid`	grid points where the density estimate is evaluated. The default is NULL.
`prior`	a list giving the prior information. The list includes the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Polya Tree prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` and `b0` are missing, see details below), `tau1` and `tau2` giving the hyperparameters for the prior distribution of the variance of the normal baseline distribution, `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the regression coefficients, optionally `M` giving the finite level to be considered, and `frstlprob` a logical variable indicating whether the first level probabilities of the PT are fixed defining a median regression model (the default is TRUE). Note that if `M` is specified, a Partially Specified Mixture of Polya trees is fitted.
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on the screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `PTlm` to print an error message and terminate if there are any incomplete observations.

Details

By default, this generic function fits a median regression model using a Scale Mixture of Polya Trees prior for the distribution of the errors (see, e.g., Lavine, 1992 and 1994, Hanson and Johnson, 2004):

$$y_i = X_i\beta + V_i, i = 1, \ldots, n$$

$$V_i|G \sim G$$
$$G|\alpha, \sigma^2 \sim PT(\Pi^{\sigma^2}, A)$$

where, the PT is centered around a $N(0, \sigma^2)$ distribution, by taking each m level of the partition Π^{σ^2} to coincide with the $k/2^m, k = 0, \ldots, 2^m$ quantile of the $N(0, \sigma^2)$ distribution. The family $A = \{\alpha_e : e \in E^*\}$, where $E^* = \bigcup_{m=1}^{\infty} E^m$ and E^m is the m-fold product of $E = \{0, 1\}$, was specified as $\alpha_{e_1 \ldots e_m} = \alpha m^2$. To complete the model specification, independent hyperpriors are assumed,

$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$
$$\sigma^{-2}|\tau_1, \tau_2 \sim Gamma(\tau_1/2, \tau_2/2)$$

Optionally, if frstlprob=FALSE (the default value is TRUE) is specified, a mean regression model is considered. In this case, the following PT prior is considered:

$$G|\alpha, \mu, \sigma^2 \sim PT(\Pi^{\mu,\sigma^2}, A)$$

where, the PT is centered around a $N(0, \mu, \sigma^2)$ distribution. In this case, the intercept term is automatically excluded from the model and the hyperparameters for the normal prior for μ must be specified. The normal prior is given by,

$$\mu|\mu_b, S_b \sim N(\mu_b, S_b)$$

The precision parameter, α, of the `PT` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

In the computational implementation of the model, Metropolis-Hastings steps are used to sample the posterior distribution of the regression coefficients and hyperparameters.

Value

An object of class `PTlm` representing the semiparametric median regression model fit. Generic functions such as `print`, `plot`, `summary`, and `anova` have methods to show the results of the fit. The results include `beta`, `mu`, `sigma2`, and the precision parameter `alpha`.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`alpha`	giving the value of the precision parameter.
`beta`	giving the value of the regression coefficients.
`mu`	giving the mean of the normal baseline distribution (If needed).
`sigma2`	giving the variance of the normal baseline distribution.
`v`	giving the value of the errors (it must be consistent with the data.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

References

Hanson, T., and Johnson, W. (2002) Modeling regression error with a Mixture of Polya Trees. Journal of the American Statistical Association, 97: 1020 - 1033.

Lavine, M. (1992) Some aspects of Polya tree distributions for statistical modelling. The Annals of Statistics, 20: 1222-11235.

Lavine, M. (1994) More aspects of Polya tree distributions for statistical modelling. The Annals of Statistics, 22: 1161-1176.

Examples

```
## Not run:
    ####################################
    # A simulated Data Set
    # (Mixture of Normals)
    ####################################

      ind<-rbinom(100,1,0.5)
      vsim<-ind*rnorm(100,1,0.15)+(1-ind)*rnorm(100,3,0.15)

      x1<-rep(c(0,1),50)
      x2<-rnorm(100,0,1)

      etasim<-x1+-1*x2
      y<-etasim+vsim


    # Initial state
      state <- NULL

    # MCMC parameters
      nburn<-5000
      nsave<-10000
      nskip<-20
      ndisplay<-100
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,
                   ndisplay=ndisplay)

    # Prior information
      prior <- list(alpha=1,beta0=rep(0,3),Sbeta0=diag(1000,3),
                    tau1=0.01,tau2=0.01,M=6)

    # Fit the model

      fit1 <- PTlm(formula=y~x1+x2,prior=prior,mcmc=mcmc,state=state,
                   status=TRUE)

    # Summary with HPD and Credibility intervals
      summary(fit1)
      summary(fit1,hpd=FALSE)

    # Plot model parameters (to see the plots gradually set ask=TRUE)
      plot(fit1)
      plot(fit1,nfigr=2,nfigc=2)

    # Table of Pseudo Contour Probabilities
      anova(fit1)


    ############################################
    # The Australian Institute of Sport's data
    # (Skew data example)
    ############################################
      data(sports)
      attach(sports)

    # Initial state
      state <- NULL

    # MCMC parameters

      nburn<-5000
      nsave<-10000
      nskip<-20
      ndisplay<-100
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,
                   ndisplay=ndisplay)

    # Prior information
      prior <- list(alpha=1,beta0=rep(0,3),Sbeta0=diag(1000,3),
                    tau1=0.01,tau2=0.01,M=8)

    # Fit the model

      fit2 <- PTlm(formula=bmi~lbm+gender,prior=prior,mcmc=mcmc,
                   state=state,status=TRUE)

    # Summary with HPD and Credibility intervals
      summary(fit2)
      summary(fit2,hpd=FALSE)

    # Plot model parameters (to see the plots gradually set ask=TRUE)
      plot(fit2)
      plot(fit2,nfigr=2,nfigc=2)

    # Table of Pseudo Contour Probabilities
      anova(fit2)


## End(Not run)
```

PTlmm — *Bayesian analysis for a semiparametric linear mixed model using a MMPT*

Description

This function generates a posterior density sample for a semiparametric linear mixed model, using a Mixture of Multivariate Polya Trees prior for the distribution of the random effects.

Usage

```
PTlmm(fixed,random,prior,mcmc,state,status,data=sys.frame(sys.parent()),
      na.action=na.fail)
```

Arguments

`fixed`	a two-sided linear formula object describing the fixed-effects part of the model, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right.
`random`	a one-sided formula of the form `~z1+...+zn | g`, with `z1+...+zn` specifying the model for the random effects and `g` the grouping variable. The random effects formula will be repeated for all levels of grouping.
`prior`	a list giving the prior information. The list include the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Polya Tree (PT) prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` and `b0` are missing, see details below), `nu0` and `tinv` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix of the normal baseline distribution, `sigma` giving the value of the covariance matrix of the centering distribution (it must be specified if `nu0` and `tinv` are missing), `mub` and `Sb` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `mu` giving the value of the mean of the centering distribution (it must be specified if `mub` and `Sb` are missing), `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the fixed effects (must be specified only if fixed effects are considered in the model), `tau1` and `tau2` giving the hyperparameters for the prior distribution of the error variance, `M` giving the finite level of the PT prior to be considered, `frstlprob` a logical variable indicating whether the first level probabilities of the PT are fixed or not (the default is FALSE), and `typepr` indicating whether the type of decomposition of the centering covariance matrix is random (1) or not (0).
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out), `nbase` giving the number scans to be performed before the parameters of the centering distribution and the precision parameter are updated (i.e., the update of this parameters is invoked only once in every `nbase` scans) (the default value is 1), `tune1`, `tune2`, and `tune3`, giving the Metropolis tuning parameter for the baseline mean, variance, and precision parameter, respectively. If `tune1`, `tune2`, or `tune3` are not specified or negative, an adpative Metropolis algorithm is performed. Finally,

	the integer `samplef` indicates whether the functional parameters must be sample (1) or not (0).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `PTlmm` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a linear mixed-effects model using a Mixture of Multivariate Polya Trees prior (see, Lavine 1992; 1994, for details about univariate PT) for the distribution of the random effects as described in Jara, Hanson and Lesaffre (2009):

$$y_i \sim N(X_i\beta_F + Z_i\beta_R + Z_i b_i, \sigma_e^2 I_{n_i}), i = 1, \ldots, n$$

$$\theta_i | G \sim G$$

$$G|\alpha, \mu, \Sigma, O \sim PT^M(\Pi^{\mu,\Sigma,O}, \mathcal{A})$$

$$\sigma_e^{-2}|\tau_1, \tau_2 \sim Gamma(\tau_1/2, \tau_2/2)$$

where, $\theta_i = \beta_R + b_i$, $\beta = \beta_F$, and O is an orthogonal matrix defining the decomposition of the centering covariance matrix. As in Hanson (2006), the PT prior is centered around a $N_d(\mu, \Sigma)$ distribution. However, we consider the class of partitions $\Pi^{\mu,\Sigma,O}$. The partitions starts with base sets that are Cartesian products of intervals obtained as quantiles from the standard normal distribution. A multivariate location-scale transformation, $\theta = \mu + \Sigma^{1/2} z$, is applied to each base set yielding the final sets. Here $\Sigma^{1/2} = T'O'$ where T is the unique upper triangular Cholesky matrix of Σ. The family $\mathcal{A} = \{\alpha_e : e \in E^*\}$, where $E^* = \bigcup_{m=0}^{M} E_d^m$, with E_d and E_d^m the d-fold product of $E = \{0, 1\}$ and the the m-fold product of E_d, respectively. The family $\mathcal{A}$ was specified as $\alpha_{e_1 \ldots e_m} = \alpha m^2$.

To complete the model specification, independent hyperpriors are assumed,

$$\alpha|a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\beta|\beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

$$\mu|\mu_b, S_b \sim N(\mu_b, S_b)$$

$$\Sigma|\nu_0, T \sim IW(\nu_0, T)$$

$$O \sim Haar(q)$$

Note that the inverted-Wishart prior is parametrized such that $E(\Sigma) = T^{-1}/(\nu_0 - q - 1)$.

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value.

The computational implementation of the model is based on the marginalization of the `PT` as descried in Jara, Hanson and Lesaffre (2009).

Value

An object of class `PTlmm` representing the linear mixed-effects model fit. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `betaR`, `betaF`, `sigma2e`, `mu`, the elements of `Sigma`, `alpha`, and `ortho`.

The function `PTrandom` can be used to extract the posterior mean of the random effects.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`alpha`	giving the value of the precision parameter
`b`	a matrix of dimension (nsubjects)*(nrandom effects) giving the value of the random effects for each subject.
`beta`	giving the value of the fixed effects.
`mu`	giving the mean of the normal baseline distributions.
`sigma`	giving the variance matrix of the normal baseline distributions.
`sigma2e`	giving the error variance.
`ortho`	giving the orthogonal matrix `H`, used in the decomposition of the covariance matrix.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

Tim Hanson <<`hansont@stat.sc.edu`>>

References

Hanson, T. (2006) Inference for Mixtures of Finite Polya Trees. Journal of the American Statistical Association, 101: 1548-1565.

Jara, A., Hanson, T., Lesaffre, E. (2009) Robustifying Generalized Linear Mixed Models using a New Class of Mixtures of Multivariate Polya Trees. Journal of Computational and Graphical Statistics, 18(4): 838-860.

Lavine, M. (1992) Some aspects of Polya tree distributions for statistical modelling. The Annals of Statistics, 20: 1222-11235.

Lavine, M. (1994) More aspects of Polya tree distributions for statistical modelling. The Annals of Statistics, 22: 1161-1176.

See Also

`PTrandom`, `PTglmm` , `PTolmm`, `DPMglmm`, `DPMlmm`, `DPMolmm`, `DPlmm` , `DPglmm`, `DPolmm`

Examples

```
## Not run:
    # School Girls Data Example
      data(schoolgirls)
      attach(schoolgirls)

    # Prior information
      prior <- list(a0=5,b0=1,
                    M=4,
                    typepr=1,
                    frstlprob=FALSE,
                    tau1=0.01,tau2=0.01,
                    nu0=4.01,
                    tinv=diag(10,2),
                    mub=rep(0,2),
                    Sb=diag(1000,2))

    # Initial state
      state <- NULL

    # MCMC parameters

      nburn <- 10000
      nsave <- 10000
      nskip <- 20
      ndisplay <- 1000
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay,
                   tune1=1.5,tune2=1.1,samplef=1)

    # Fitting the model

      fit1 <- PTlmm(fixed=height~1,random=~age|child,prior=prior,mcmc=mcmc,
                    state=state,status=TRUE)
      fit1

    # Summary with HPD and Credibility intervals
      summary(fit1)
      summary(fit1,hpd=FALSE)

    # Plot model parameters (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE)
      plot(fit1,ask=FALSE,nfigr=2,nfigc=2)

    # Plot an specific model parameter (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE,nfigr=1,nfigc=2,param="sigma-(Intercept)")

    # Random effects information
      PTrandom(fit1)

    # Predictive marginal and joint distributions
      plot(PTrandom(fit1,predictive=TRUE))
```

```
## End(Not run)
```

`PTmeta` *Bayesian analysis for a semiparametric linear mixed effects meta-analysis model using a MPT*

Description

This function generates a posterior density sample for a semiparametric linear mixed effects meta-analysis model using a Polya Tree or a Mixture of Polya Trees prior for the distribution of the random effects.

Usage

```
PTmeta(formula,prior,mcmc,state,status,data=sys.frame(sys.parent()),
       na.action=na.fail)
```

Arguments

`formula`	a two-sided linear formula object describing the fixed-effects part of the model, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right. Both effect and variance must be included in the LHS of the formula object
`prior`	a list giving the prior information. The list include the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Polya tree prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` and `b0` are missing, see details below), `tau1` and `tau2` giving the hyperparameters for the prior distribution of the variance of the centering distribution, `sigma` giving the value of the variance of the centering distribution (it must be specified if `tau1` and `tau2` are missing), `mub` and `Sb` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `mu` giving the value of the mean of the centering distribution (it must be specified if `mub` and `Sb` are missing), and `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the fixed effects (must be specified only if fixed effects are considered in the model), `M` giving the finite level of the PT prior to be considered, and `frstlprob` a logical variable indicating whether the first level probabilities of the PT are fixed or not (the default is FALSE) (see, details).
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.

`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `PTmeta` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a semiparametric linear mixed effects meta-analysis model using a Polya tree prior on the distribution (see, Lavine (1992; 1994) and Hanson (2006) for details about PT) on the distribution of the random effects:

$$y_i \sim N(\theta_i + X_i\beta, \sigma_{ei}^2), i = 1, \ldots, n$$

$$\theta_i | G \sim G$$

$$G | \alpha, \mu, \sigma \sim PT(\Pi^{\mu,\sigma}, A)$$

where the PT prior is centered around a $N(\mu, \sigma^2)$ distribution. If `frstlprob` is equal to TRUE, $\mu = 0$ and a median zero PT prior is considered (see, Branscum and Hanson, 2008).

To complete the model specification, independent hyperpriors are assumed,

$$\alpha | a_0, b_0 \sim Gamma(a_0, b_0)$$

$$\beta | \beta_0, S_{\beta_0} \sim N(\beta_0, S_{\beta_0})$$

$$\mu | \mu_b, S_b \sim N(\mu_b, S_b)$$

$$\sigma^{-2} | \tau_1, \tau_2 \sim Gamma(\tau_1/2, \tau_2/2)$$

The precision parameter, α, of the `PT` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value.

The computational implementation of the model is based on the marginalization of the `PT` and on the MCMC algorihtms described in Hanson (2006) and Jara, Hanson and Lesaffre (2009).

The average effect is sampled using the method of composition described in Jara, Hanson and Lesaffre (2009).

Value

An object of class `PTmeta` representing the linear mixed-effects model fit. Generic functions such as `print`, `plot`, `summary`, and `anova` have methods to show the results of the fit. The results include `beta`, `mu`, `sigma2`, and `alpha`.

The function `PTrandom` can be used to extract the posterior mean of the random effects.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`alpha`	giving the value of the precision parameter
`b`	a vector of dimension (nsubjects) giving the value of the random effects for each subject.
`beta`	giving the value of the fixed effects.
`mu`	giving the mean of the normal baseline distributions.
`sigma2`	giving the variance of the normal baseline distributions.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

References

Branscum, A. and Hanson, T. (2008) Bayesian nonparametric meta-analysis using Polya tree mixture models. Biometrics, 64: 825-833.

Christensen, R., Hanson, T. Jara, A.. 2008. Parametric Nonparametric Statistics: An Introduction to Mixtures of Finite Polya Trees Models. The American Statistician, 62: 296-306.

Hanson, T. (2006) Inference for Mixtures of Finite Polya Trees. Journal of the American Statistical Association, 101: 1548-1565.

Jara, A., Hanson, T., Lesaffre, E. (2009) Robustifying Generalized Linear Mixed Models using a New Class of Mixtures of Multivariate Polya Trees. Journal of Computational and Graphical Statistics, 18(4): 838-860.

Lavine, M. (1992) Some aspects of Polya tree distributions for statistical modelling. The Annals of Statistics, 20: 1222-11235.

Lavine, M. (1994) More aspects of Polya tree distributions for statistical modelling. The Annals of Statistics, 22: 1161-1176.

See Also

`PTrandom`, `DPMmeta` , `DPMmeta` , `DPlmm` , `DPglmm` , `DPolmm` , `DPMlmm` , `DPMglmm` , `DPMolmm`

Examples

```
## Not run:

    ##################################################################
    # Data on the effectiveness of silver sulfadiazine coating
    # on venous catheters for preventing bacterial colonisation of
    # the catheter and bloodstream infection.
    # Veenstra D et al (1998) "Efficacy of Antiseptic Impregnated
    # Central Venous Catheters in Preventing Nosocomial Infections:
    # A Meta-analysis" JAMA 281:261-267.
    #
    # Note that -Inf and Inf have been replaced by NA.
    ##################################################################

      studies <- c("Tennenberg","Maki","vanHeerden",
                   "Hannan","Bach(a)","Bach(b)",
                    "Heard","Collins","Ciresi","Ramsay",
                    "Trazzera","George")

      logOR <- c(-1.5187189,-0.7136877,-1.3217558,-0.1910552,
                  NA,-2.2005195,-0.5057461,-2.3538784,-0.3643810,
                  -0.5371429,-0.7608058,-2.1400662)

      varlogOR <- c(0.4157541,0.2632550,0.6739189,0.3727788,NA,
                    0.7623470,0.2306169,0.7477891,0.3645463,0.2291839,
                    0.3561542,0.5190489)^2

      names(logOR) <- studies
      names(varlogOR) <- studies
      y <- cbind(logOR,varlogOR)
      colnames(y) <- c("logOR","varlogOR")


    # Initial state
      state <- NULL

    # MCMC parameters

      nburn<-20000
      nsave<-10000
      nskip<-20
      ndisplay<-100
      mcmc <- list(nburn=nburn,
                   nsave=nsave,
                   nskip=nskip,
                   ndisplay=ndisplay)

    # Prior information 1: non-median zero PT

      prior1<-list(alpha=1,
                   tau1=20,
                   tau2=10,
                   mub=0,
                   Sb=100,
                   M=4)

    # Prior information 2: median zero PT

      prior2<-list(alpha=1,
                   tau1=20,
                   tau2=10,
                   mub=0,
                   Sb=100,
                   M=4,
                   frstlprob=TRUE,
                   Sbeta0=diag(1000,1),
                   beta0=rep(0,1))
```

```
    # Fitting the models

      fit1<-PTmeta(formula=y~1,prior=prior1,mcmc=mcmc,
                   state=state,status=TRUE)
      fit1


      fit2<-PTmeta(formula=y~1,prior=prior2,mcmc=mcmc,
                   state=state,status=TRUE)
      fit2

    # Summary with HPD and Credibility intervals
      summary(fit1)
      summary(fit1,hpd=FALSE)

      summary(fit2)
      summary(fit2,hpd=FALSE)

    # Plot model parameters (to see the plots gradually set ask=TRUE)
      plot(fit1,ask=FALSE)
      plot(fit1,ask=FALSE,nfigr=2,nfigc=2)

      plot(fit2,ask=FALSE)
      plot(fit2,ask=FALSE,nfigr=2,nfigc=2)

## End(Not run)
```

PTolmm *Bayesian analysis for a semiparametric ordinal linear mixed model using a MMPT*

Description

This function generates a posterior density sample for a semiparametric ordinal linear mixed model, using a Mixture of Multivariate Polya Trees prior for the distribution of the random effects.

Usage

```
PTolmm(fixed,random,prior,mcmc,state,status,data=sys.frame(sys.parent()),
      na.action=na.fail)
```

Arguments

`fixed`	a two-sided linear formula object describing the fixed-effects part of the model, with the response on the left of a ~ operator and the terms, separated by + operators, on the right.
`random`	a one-sided formula of the form `~z1+...+zn	g`, with `z1+...+zn` specifying the model for the random effects and `g` the grouping variable. The random effects formula will be repeated for all levels of grouping.

`prior`	a list giving the prior information. The list include the following parameter: `a0` and `b0` giving the hyperparameters for prior distribution of the precision parameter of the Polya Tree (PT) prior, `alpha` giving the value of the precision parameter (it must be specified if `a0` and `b0` are missing, see details below), `nu0` and `tinv` giving the hyperparameters of the inverted Wishart prior distribution for the scale matrix of the normal baseline distribution, `sigma` giving the value of the covariance matrix of the centering distribution (it must be specified if `nu0` and `tinv` are missing), `mub` and `Sb` giving the hyperparameters of the normal prior distribution for the mean of the normal baseline distribution, `mu` giving the value of the mean of the centering distribution (it must be specified if `mub` and `Sb` are missing), `beta0` and `Sbeta0` giving the hyperparameters of the normal prior distribution for the fixed effects (must be specified only if fixed effects are considered in the model), `M` giving the finite level of the PT prior to be considered, `frstlprob` a logical variable indicating whether the first level probabilities of the PT are fixed or not (the default is FALSE), and `typepr` indicating whether the type of decomposition of the centering covariance is random (1) or not (0).
`mcmc`	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out), `nbase` giving the number scans to be performed before the parameters of the centering distribution and the precision parameter are updated (i.e., the update of this parameters is invoked only once in every `nbase` scans) (the default value is 1), `tune1`, `tune2`, and `tune3`, giving the Metropolis tuning parameter for the baseline mean, variance, and precision parameter, respectively. If `tune1`, `tune2`, or `tune3` are not specified or negative, an adpative Metropolis algorithm is performed. Finally, the integer `samplef` indicates whether the functional parameters must be sample (1) or not (0).
`state`	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `PTolmm` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits an ordinal linear mixed-effects model with a probit link and a Mixture of Multivariate Polya Trees prior (see, Lavine 1992; 1994, for details about univariate PT) for the distribution of the random effects as described in Jara, Hanson and Lessaffre (2009):

$$Y_{ij} = k, \text{ if } \gamma_{k-1} \leq W_{ij} < \gamma_k, k = 1, \ldots, K$$

$$W_{ij} \mid \beta_F, \beta_R, b_i \sim N(X_{ij}\beta_F + Z_{ij}\beta_R + Z_{ij}b_i, 1), i = 1, \ldots, N, j = 1, \ldots, n_i$$

$$\theta_i|G \sim G$$
$$G|\alpha,\mu,\Sigma,O \sim PT^M(\Pi^{\mu,\Sigma,O},\mathcal{A})$$

where, $\theta_i = \beta_R + b_i$, $\beta = \beta_F$, and O is an orthogonal matrix defining the decomposition of the centering covariance matrix. As in Hanson (2006), the PT prior is centered around a $N_d(\mu,\Sigma)$ distribution. However, we consider the class of partitions $\Pi^{\mu,\Sigma,O}$. The partitions starts with base sets that are Cartesian products of intervals obtained as quantiles from the standard normal distribution. A multivariate location-scale transformation, $\theta = \mu + \Sigma^{1/2}z$, is applied to each base set yielding the final sets. Here $\Sigma^{1/2} = T'O'$ where T is the unique upper triangular Cholesky matrix of Σ. The family $\mathcal{A} = \{\alpha_e : e \in E^*\}$, where $E^* = \bigcup_{m=0}^{M} E_d^m$, with E_d and E_d^m the d-fold product of $E = \{0,1\}$ and the the m-fold product of E_d, respectively. The family $\mathcal{A}$ was specified as $\alpha_{e_1 \ldots e_m} = \alpha m^2$.

To complete the model specification, independent hyperpriors are assumed,

$$\alpha|a_0,b_0 \sim Gamma(a_0,b_0)$$
$$\beta|\beta_0,S_{\beta_0} \sim N(\beta_0,S_{\beta_0})$$
$$\mu|\mu_b,S_b \sim N(\mu_b,S_b)$$
$$\Sigma|\nu_0,T \sim IW(\nu_0,T)$$
$$O \sim Haar(q)$$

A uniform prior is used for the cutoff points. Note that the inverted-Wishart prior is parametrized such that $E(\Sigma) = T^{-1}/(\nu_0 - q - 1)$.

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0,b_0)$, or fixed at some particular value.

The computational implementation of the model is based on the marginalization of the `PT` as descried in Jara, Hanson and Lessaffre (2009).

Value

An object of class `PTolmm` representing the linear mixed-effects model fit. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit. The results include `betaR`, `betaF`, `mu`, the elements of `Sigma`, `alpha`, and `ortho`.

The function `PTrandom` can be used to extract the posterior mean of the random effects.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`alpha`	giving the value of the precision parameter
`b`	a matrix of dimension (nsubjects)*(nrandom effects) giving the value of the random effects for each subject.
`cutoff`	a real vector defining the cutoff points. Note that the first cutoff must be fixed at 0 in this function.
`beta`	giving the value of the fixed effects.

`mu`	giving the mean of the normal baseline distributions.
`sigma`	giving the variance matrix of the normal baseline distributions.
`ortho`	giving the orthogonal matrix `H`, used in the decomposition of the covariance matrix.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

Tim Hanson <<`hansont@stat.sc.edu`>>

References

Hanson, T. (2006) Inference for Mixtures of Finite Polya Trees. Journal of the American Statistical Association, 101: 1548-1565.

Jara, A., Hanson, T., Lesaffre, E. (2009) Robustifying Generalized Linear Mixed Models using a New Class of Mixtures of Multivariate Polya Trees. Journal of Computational and Graphical Statistics, 18(4): 838-860.

Lavine, M. (1992) Some aspects of Polya tree distributions for statistical modelling. The Annals of Statistics, 20: 1222-11235.

Lavine, M. (1994) More aspects of Polya tree distributions for statistical modelling. The Annals of Statistics, 22: 1161-1176.

See Also

`PTrandom`, `PTlmm` , `PTglmm`, `DPMglmm`, `DPMlmm`, `DPMolmm`, `DPlmm` , `DPglmm`, `DPolmm`

Examples

```
## Not run:

    # Schizophrenia Data
      data(psychiatric)
      attach(psychiatric)

    # Prior information
      prior <- list(M=4,
                    frstlprob=FALSE,
                    alpha=1,
                    nu0=4.01,
                    tinv=diag(1,1),
                    mub=rep(0,1),
                    Sb=diag(100,1),
                    beta0=rep(0,3),
                    Sbeta0=diag(1000,3))

    # MCMC parameters
      mcmc <- list(nburn=10000,
                   nsave=10000,
                   nskip=20,
                    ndisplay=100,
                    samplef=1)

    # Initial state
      state <- NULL

    # Fitting the model
      fit1 <- PTolmm(fixed=imps79o~sweek+tx+sweek*tx,random=~1|id,prior=prior,
                      mcmc=mcmc,state=state,status=TRUE)
      fit1

    # Summary with HPD and Credibility intervals
      summary(fit1)
      summary(fit1,hpd=FALSE)

    # Plot model parameters
      plot(fit1)

    # Plot an specific model parameter
      plot(fit1,ask=FALSE,nfigr=1,nfigc=2,param="sigma-(Intercept)")

    # Extract random effects
      PTrandom(fit1)

    # Extract predictive information of random effects
      aa<-PTrandom(fit1,predictive=TRUE)
      aa

    # Predictive marginal and joint distributions
      plot(aa)

## End(Not run)
```

PTrandom *Extracts Random Effects*

Description

Extracts random effects from PTglmm objects: PTlmm, PTolmm, and PTglmm.

Usage

```
PTrandom(object,centered=FALSE,predictive=FALSE,ngrid=1000,gridl=NULL)
```

Arguments

`object`	PT fitted model object from which random effects estimates can be extracted.
`centered`	logical variable indicating whether the random effects should be extracted centered, `bi`, or uncentered `thetai`. This option cannot be only used to get the density estimates.

`predictive`	logical variable indicating whether actual or predictive information of the random effects should be extracted.
`ngrid`	number of grid points where the density estimate is evaluated. This is only used if dimension of the random effects is lower or equal than 2 and if predictive=TRUE. The default value is 1000.
`gridl`	The limits of the interval or rectangle covered by the grid as c(xl,xu) or c(xl, xu, yl, yu), respectively. If not specified the grid is defined automatically. This is only used if dimension of the random effects is lower or equal than 2 and if predictive=TRUE.

Author(s)

Alejandro Jara <<`atjara@uc.cl`>>

Tim Hanson <<`hansont@stat.sc.edu`>>

Examples

```
## Not run:
    # School Girls Data Example
      data(schoolgirls)
      attach(schoolgirls)

    # Prior information
      prior<-list(alpha=1,
                  M=4,
                  tau1=0.01,tau2=0.01,
                  nu0=4.01,
                  tinv=diag(10,2),
                  mub=rep(0,2),
                  Sb=diag(1000,2))

    # Initial state
      state <- NULL

    # MCMC parameters

      nburn<-5000
      nsave<-5000
      nskip<-0
      ndisplay<-100
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay,
                    tune1=1.5,tune2=1.1)

    # Fitting the model

      fit1<-PTlmm(fixed=height~1,random=~age|child,prior=prior,mcmc=mcmc,
                  state=state,status=TRUE)
      fit1


    # Extract random effects
```

```
      PTrandom(fit1)

      plot(PTrandom(fit1))

    # Extract predictive information of random effects

      PTrandom(fit1,predictive=TRUE)
      plot(PTrandom(fit1,predictive=TRUE,gridl=c(75,89,3.8,7.5)))

## End(Not run)
```

PTsampler *Polya Tree sampler function*

Description

This function allows a user to generate a sample from a user-defined unormalized continuos distribution using the Polya tree sampler algorithm.

Usage

```
PTsampler(ltarget,dim.theta,mcmc=NULL,support=NULL,pts.options=NULL,
  status=TRUE,state=NULL)
```

Arguments

`ltarget`	a function giving the log of the target density.
`dim.theta`	an integer indicating the dimension of the target density.
`mcmc`	an optional list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out). Default values are 1000, 1000, and 100 for `nburn`, `nsave`, and `ndisplay`, respectively.
`support`	an optional matrix, of dimension dim.theta * npoints, giving the initial support points. By default the function generates 400 support points from a dim.theta normal distribution with mean 0 and diagonal covariance matrix with 1000 in the diagonal.
`pts.options`	an optional list of giving the parameters needed for the PTsampler algorithm. The list must include: `nlevel` (an integer giving the number of levels of the finite Polya tree approximation; default=5), `tune1` (a double precision variable representing the standard deviation of the log-normal candidate distribution for the precision parameter of the Polya tree; default=1), `delta` (a double precision number indicating the maximum distance between the target and the approximation; default=0.2), `max.warmup` (an integer giving the maximum number of

	steps allowed for the warm-up phase; default=50000), `minc` (a double precision variable giving the minimum value allowed for the precision parameter of the Polya tree approximation; default=1), `cpar0` (a double precision variable giving the initial value for the precision parameter of the Polya tree approximation; default=1000), and `nadd` (an integer variable giving the number of warm-up steps after convergence; default=1000).
`status`	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.
`state`	a list giving the starting points for the MCMC algorithm. The list must include: `theta` (a vector of dimension dim.theta of parameters), `u` (a Polya tree decomposition matrix), `uinv` (a matrix giving the inverse of the decompositon matrix), `cpar` (giving the value of the Polya tree precision parameter), `support` (a matrix giving the final support points), `dim.theta` (an integer giving the dimension of the problem), and `L1` (a double precision number giving the final convergence criterion value).

Details

PTsampler produces a sample from a user-defined multivariate distribution using the Polya tree sampler algorithm. The algorithm constructs an independent proposal based on an approximation of the target density. The approximation is built from a set of support points and the predictive density of a finite multivariate Polya tree. In an initial warm-up phase, the support points are iteratively relocated to regions of higher support under the target distribution to minimize the distance between the target distribution and the Polya tree predictive distribution. In the sampling phase, samples from the final approximating mixture of finite Polya trees are used as candidates which are accepted with a standard Metropolis-Hastings acceptance probability. We refer to Hanson, Monteiro, and Jara (2011) for more details on the Polya tree sampler.

Value

An object of class `PTsampler` representing the MCMC sampler. Generic functions such as `print`, `plot`, and `summary` have methods to show the results of the fit.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values.

The object `thetsave` in the output list `save.state` contains the samples from the target density.

Author(s)

Alejandro Jara <`<atjara@uc.cl>`>

Tim Hanson <`<hansont@stat.sc.edu>`>

References

Hanson, T., Monteiro, J.V.D, and Jara, A. (2011) The Polya Tree Sampler: Toward Efficient and Automatic Independent Metropolis-Hastings Proposals. Journal of Computational and Graphical Statistics, 20: 41-62.

See Also

`PTdensity`

Examples

```
## Not run:

################################
# EXAMPLE 1 (Dog Bowl)
################################

# Target density

  target <- function(x,y)
  {
     out <- (-3/2)*log(2*pi)-0.5*(sqrt(x^2+y^2)-10)^2-
            0.5*log(x^2+y^2)
     exp(out)
  }

  ltarget <- function(x)
  {
     out <- -0.5*((sqrt(x[1]^2+x[2]^2)-10)^2)-
             0.5*log(x[1]^2+x[2]^2)
     out
  }

# MCMC

  mcmc <- list(nburn=5000,
               nsave=10000,
               ndisplay=500)

# Initial support points (optional)

  support <- cbind(rnorm(300,15,1),rnorm(300,15,1))

# Scanning the posterior

  fit <- PTsampler(ltarget,dim.theta=2,mcmc=mcmc,support=support)

  fit
  summary(fit)
  plot(fit,ask=FALSE)

# Samples saved in
# fit$save.state$thetasave
# Here is an example of how to use them

  par(mfrow=c(1,2))
  plot(acf(fit$save.state$thetasave[,1],lag=100))
  plot(acf(fit$save.state$thetasave[,1],lag=100))
```

```
# Plotting resulting support points

  x1 <- seq(-15,15,0.2)
  x2 <- seq(-15,15,0.2)
  z <- outer(x1,x2,FUN="target")
  par(mfrow=c(1,1))
  image(x1,x2,z,xlab=expression(theta[1]),ylab=expression(theta[2]))
  points(fit$state$support,pch=19,cex=0.25)

# Plotting the samples from the target density

  par(mfrow=c(1,1))
  image(x1,x2,z,xlab=expression(theta[1]),ylab=expression(theta[2]))
  points(fit$save.state$thetasave,pch=19,cex=0.25)

# Re-starting the chain from the last sample

  state <- fit$state
  fit <- PTsampler(ltarget,dim.theta=2,mcmc=mcmc,
                   state=state,status=FALSE)


###############################
# EXAMPLE 2 (Ping Pong Paddle)
###############################

  bivnorm1 <- function(x1,x2)
  {
       eval <- (x1)^2+(x2)^2
       logDET <-  0
       logPDF <- -(2*log(2*pi)+logDET+eval)/2
       out <- exp(logPDF)
       out
  }

  bivnorm2 <- function(x1,x2)
  {
       mu <- c(-3,-3)
       sigmaInv <- matrix(c(5.263158,-4.736842,
                           -4.736842,5.263158),
                            nrow=2,ncol=2)
       eval <- (x1-mu[1])^2*sigmaInv[1,1]+
               2*(x1-mu[1])*(x2-mu[2])*sigmaInv[1,2]+
               (x2-mu[2])^2*sigmaInv[2,2]
       logDET <-  -1.660731
       logPDF <- -(2*log(2*pi)+logDET+eval)/2
       out <- exp(logPDF)
       out
  }

  bivnorm3 <- function(x1,x2)
  {
       mu <- c(2,2)
       sigmaInv <- matrix(c(5.263158,4.736842,
                            4.736842,5.263158),
                            nrow=2,ncol=2)
       eval <- (x1-mu[1])^2*sigmaInv[1,1]+
               2*(x1-mu[1])*(x2-mu[2])*sigmaInv[1,2]+
               (x2-mu[2])^2*sigmaInv[2,2]
       logDET <-  -1.660731
       logPDF <- -(2*log(2*pi)+logDET+eval)/2
       out <- exp(logPDF)
       out
  }

  target <- function(x,y)
  {
     out <- 0.34*bivnorm1(x,y)+
    0.33*bivnorm2(x,y)+
    0.33*bivnorm3(x,y)
     out
  }

  ltarget <- function(theta)
  {
     out <- 0.34*bivnorm1(x1=theta[1],x2=theta[2])+
    0.33*bivnorm2(x1=theta[1],x2=theta[2])+
    0.33*bivnorm3(x1=theta[1],x2=theta[2])
     log(out)
  }


# MCMC

  mcmc <- list(nburn=5000,
               nsave=10000,
               ndisplay=500)

# Initial support points (optional)

  support <- cbind(rnorm(300,6,1),rnorm(300,6,1))

# Scanning the posterior

  fit <- PTsampler(ltarget,dim.theta=2,mcmc=mcmc,support=support)

  fit
  summary(fit)
  plot(fit,ask=FALSE)

# Samples saved in
# fit$save.state$thetasave
# Here is an example of how to use them

  par(mfrow=c(1,2))
  plot(acf(fit$save.state$thetasave[,1],lag=100))
  plot(acf(fit$save.state$thetasave[,1],lag=100))

# Plotting resulting support points

  x1 <- seq(-6,6,0.05)
  x2 <- seq(-6,6,0.05)
  z <- outer(x1,x2,FUN="target")
  par(mfrow=c(1,1))
  image(x1,x2,z,xlab=expression(theta[1]),ylab=expression(theta[2]))
  points(fit$state$support,pch=19,cex=0.25)

# Plotting the samples from the target density

  par(mfrow=c(1,1))
  image(x1,x2,z,xlab=expression(theta[1]),ylab=expression(theta[2]))
  points(fit$save.state$thetasave,pch=19,cex=0.25)



## End(Not run)
```

`rats` *Rats*

Description

This example is taken from section 6 of Gelfand and Smith (1990), and concerns 30 young rats whose weights were measured weekly for five weeks.

Usage

```
data(rats)
```

Format

A data frame with 150 observations on the following 3 variables.

`weight` a numeric vector giving the weight of the rat

`day` a numeric vector giving the day of the weight evaluation

`rat` an ordered factor giving a unique identifier for the subject in the study

Source

Gelfand, A.E. (with S. Hills, A. Racine-Poon and A.F.M. Smith) 1990. Illustration of Bayesian Inference in Normal Data Models Using Gibbs Sampling. Journal Amer. Stat. Assoc., 85, 972-985.

References

Gelfand, A.E. (with S. Hills, A. Racine-Poon and A.F.M. Smith) 1990. Illustration of Bayesian Inference in Normal Data Models Using Gibbs Sampling. Journal Amer. Stat. Assoc., 85, 972-985.

Examples

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

`rolling` *Rolling Thumbtacks Data*

Description

This data was generated by Beckett and Diaconis (1994). They generate binary strings from rools of common thumbtacks. A 1 was recorded if the tack landed point up and 0 was recorder if the tack landed point down. All tacks started point down. Each tack was flicked 9 times. The data consit of 320 9-tuples. The actual data arose from 16 "flickers" and 10 surfaces. Following Liu (1996), we treat the data as though they came from 320 different tacks.

Usage

```
data(rolling)
```

Format

A data frame with 320 observations on the following 2 variables.

`y1` a numeric vector giving the number of tacks landed point up.

`y2` a numeric vector giving the number of trials.

Source

Beckett, L. and Diaconis. P. (1994). Spectral analysis fro discrete longitudinal data. Adv. Math., 103: 107-128.

References

Liu, J.S. (1996). Nonparametric Hierarchical Bayes via Sequential Imputations. The Annals of Statistics, 24: 911-930.

Examples

```
data(rolling)
```

schoolgirls — *The Heights of Schoolgirls*

Description

This data set consider growth information of 20 preadolescent schoolgirls reported by Goldstein (1979, Table 4.3, p. 101). The height of girls was measured on a yearly basis from age 6 to 10. The measurements are given at exact years of age, some having been previously adjusted to these. Further, the girls were classified according to the height of their mother into three categories: short mothers, medium mothers and tall mothers.

Usage

```
data(schoolgirls)
```

Format

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

`height` a numeric vector giving the height in cm

`child` an ordered factor giving a unique identifier for the subject in the study

`age` a numeric vector giving the age of the child in years

`group` a factor with levels 1 (short), 2 (medium), and 3 (tall) giving the mother category

Details

Measurements reported by Goldstein(1979) for one of the girls (child 5) were 114.5, 112.0, 126.4, 131.2, and 135.0. In this data set, the second measurement was replaced by 122.0.

Source

Goldstein, H. (1979) The Design and Analysis of Longitudinal Studies. London: Academic Press.

References

Verbeke, G., and Molenberghs, G. (2000) Linear Mixed Models for Longitudinal Data. New York: Springer-Verlag.(Section 12.7)

Examples

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

seizures *Epileptic seizures*

Description

This data set consider information from a clinical trial of 59 epileptics, reported by Thall and Vail (1990). For each patient, the number of epileptic seizures was recorded during a baseline period of eight weeks. Patients were then randomized to treatment with the anti-epileptic drug progabide, or to aplacebo in addition to standard chemotherapy. The number of seizures was then recorded in four consecutive two-weeks intervals.

Usage

```
data(seizures)
```

Format

A data frame with 295 observations on the following 5 variables.

`id` an ordered factor giving a unique identifier for the subject in the study.

`seize` a numeric vector giving the number of epileptic seizures.

`visit` a numeric vector giving the number of the visit, 0=baseline, and 1,2,3, and 4 for the four consecutive two-weeks intervals.

`trt` a numeric vector giving the treatment group.

`age` a numeric vector giving the age at the entry.

Source

Thall, P.F., and Vail, S.C. (1990) Some covariance models for longitudinal count data with ovserdispersion, Biometrics, 46: 657-671.

References

Diggle, P.J., Liang, K-Y., and Zeger, S.L. (1994) Analysis of longitudinal data. Oxford: Clarendon Press.

Examples

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

`sports` *The Australian Athletes Data*

Description

This data set consider information from 202 elite Australian athletes who trained at the Australian Institute of Sport. The members of the sample participate in a number of different sports and are about equally split between men and women.

Usage

```
data(sports)
```

Format

A data frame with 202 observations on the following 13 variables.

`gender` a factor with levels `female` and `male`

`sport` a factor with levels `B_Ball`, `Field`, `Gym`, `Netball`, `Row`, `Swim`, `T_400m`, `T_Sprnt`, `Tennis`, and `W_Polo`

`rcc` a numeric vector giving the red cell count

`wcc` a numeric vector giving the white cell count

`Hc` a numeric vector giving the Hematocrit

`Hg` a numeric vector giving the Hemoglobin level

`Fe` a numeric vector giving the plasma ferritin concentration

`bmi` a numeric vector giving the body mass index, weight/(height)**2

`ssf` a numeric vector giving the sum of skin folds

`Bfat` a numeric vector giving the body fat percentage

`lbm` a numeric vector giving the lean body mass

`Ht` a numeric vector giving the height (cm)

`Wt` a numeric vector giving the weight (Kg)

Source

Cook and Weisberg (1994), An Introduction to Regression Graphics. John Wiley & Sons, New York.

References

Cook and Weisberg (1994), An Introduction to Regression Graphics. John Wiley & Sons, New York.

Examples

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

TDPdensity *Semiparametric Bayesian density estimation using DP Mixtures of Triangular Distributions*

Description

This function generates a posterior density sample for a Triangular-Dirichlet model.

Usage

```
TDPdensity(y,support=3,transform=1,ngrid=1000,prior,mcmc,state,status,
          data=sys.frame(sys.parent()),na.action=na.fail)
```

Arguments

y	a vector giving the data from which the density estimate is to be computed.
support	an integer number giving the support of the random density, 1=[0,1], 2=(0, +Inf], and 3=(-In,+Inf). Depending on this, the data is transformed to lie in the [0,1] interval.
transform	an integer number giving the type of transformation to be considered, 1=Uniform, 2=Normal,3=Logistic,4=Cauchy. The types 2-4 can be only used when the support is the real line.
ngrid	number of grid points where the density estimate is evaluated. This is only used if dimension of `y` is lower or equal than 2. The default value is 1000.
prior	a list giving the prior information. The list includes the following parameter: `aa0` and `ab0` giving the hyperparameters for prior distribution of the precision parameter of the Dirichlet process prior, `alpha` giving the value of the precision parameter (it must be specified if `aa0` is missing, see details below), `a0` and `b0` giving the parameters of the beta centering distribution of the DP prior, and `kmax` giving the maximum value of the discrete uniform prior for number of components in the Mixture of Triangular distributions. Optionally, when the support of the data is the real line and the parametric transformation 2-4 are considered, the location `mu` and the scale parameter `sigma2` can be included here. If not, they are taked as the mean and the variance of the data, respectively.
mcmc	a list giving the MCMC parameters. The list must include the following integers: `nburn` giving the number of burn-in scans, `nskip` giving the thinning interval, `nsave` giving the total number of scans to be saved, and `ndisplay` giving the number of saved scans to be displayed on screen (the function reports on the screen when every `ndisplay` iterations have been carried out).
state	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
status	a logical variable indicating whether this run is new (`TRUE`) or the continuation of a previous analysis (`FALSE`). In the latter case the current value of the parameters must be specified in the object `state`.

`data`	data frame.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `TDPdensity` to print an error message and terminate if there are any incomplete observations.

Details

This generic function fits a Triangular-Dirichlet model for density estimation:

$$y_i|G \sim G, i = 1, \ldots, n$$

$$G|kmax, \alpha, G_0 \sim TDP(kmax, \alpha G_0)$$

where, y_i is the transformed data to lie in [0,1], `kmax` is the upper limit of the discrete uniform prior for the number of components in the Mixture of Triangular distributions, α is the total mass parameter of the Dirichlet process component, and G_0 is the centering distribution of the DP. The centering distribution corresponds to a $G_0 = Beta(a_0, b_0)$ distribution.

Note that our representation is different to the Mixture of Triangular distributions proposed by Perron and Mengersen (2001). In this function we consider random weights following a Dirichlet prior and we exploit the underlying DP structure. By so doing, we avoid using Reversible-Jumps algorithms.

The precision or total mass parameter, α, of the `DP` prior can be considered as random, having a `gamma` distribution, $Gamma(a_0, b_0)$, or fixed at some particular value. When α is random the method described by Escobar and West (1995) is used. To let α to be fixed at a particular value, set a_0 to NULL in the prior specification.

Value

An object of class `TDPdensity` representing the Triangular-Dirichlet model fit. Generic functions such as `print`, `summary`, and `plot` have methods to show the results of the fit. The results include the degree of the polynomial `k`, `alpha`, and the number of clusters.

The MCMC samples of the parameters and the errors in the model are stored in the object `thetasave` and `randsave`, respectively. Both objects are included in the list `save.state` and are matrices which can be analyzed directly by functions provided by the coda package.

The list `state` in the output object contains the current value of the parameters necessary to restart the analysis. If you want to specify different starting values to run multiple chains set `status=TRUE` and create the list state based on this starting values. In this case the list `state` must include the following objects:

`ncluster`	an integer giving the number of clusters.
`yclus`	a real vector giving the `y` latent variables of the clusters (only the first `ncluster` are considered to start the chain).
`ss`	an interger vector defining to which of the `ncluster` clusters each observation belongs.
`alpha`	giving the value of the precision parameter.
`k`	giving the number of components in the Mixture of Triangular distriutions.

Author(s)

Alejandro Jara <<atjara@uc.cl>>

References

Escobar, M.D. and West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association, 90: 577-588.

Perron, F. and Mengersen, K. (2001) Bayesian Nonparametric Modeling Using Mixtures of Triangular Distributions. Biometrics, 57(2): 518-528.

See Also

DPdensity, PTdensity, BDPdensity

Examples

```
## Not run:
    # Data
      data(galaxy)
      galaxy<-data.frame(galaxy,speeds=galaxy$speed/1000)
      attach(galaxy)

    # Initial state
      state <- NULL

    # MCMC parameters

      nburn<-1000
      nsave<-10000
      nskip<-10
      ndisplay<-100
      mcmc <- list(nburn=nburn,nsave=nsave,nskip=nskip,ndisplay=ndisplay)

    # Prior
      prior<-list(aa0=2.01,
                  ab0=0.01,
                  kmax=50,
                  a0=1,
                  b0=1)

    # Fitting the model

      fit<-TDPdensity(y=speeds,prior=prior,mcmc=mcmc,state=state,status=TRUE)

      plot(fit)



## End(Not run)
```

`toenail` *Toenail data*

Description

The toenail data come from a Multicenter study comparing two oral treatments for toenail infection. Patients were evaluated for the degree of separation of the nail. Patients were randomized into two treatments and were followed over seven visits - four in the first year and yearly thereafter. The patients have not been treated prior to the first visit so this should be regarded as the baseline.

Usage

```
data(toenail)
```

Format

A data frame with 1908 observations on the following 5 variables.

`ID` a numeric vector giving the ID of patient

`outcome` a numeric vector giving the response (0=none or mild seperation, 1=moderate or severe)

`treatment` a numeric vector giving the treatment gropup

`month` a numeric vector giving the time of the visit (not exactly monthly intervals hence not round numbers)

`visit` a numeric vector giving the number of the visit

Source

De Backer, M., De Vroey, C., Lesaffre, E., Scheys, I., and De Keyser, P. (1998). Twelve weeks of continuous oral therapy for toenail onychomycosis caused by dermatophytes: A double-blind comparative trial of terbinafine 250 mg/day versus itraconazole 200 mg/day. Journal of the American Academy of Dermatology, 38, 57-63.

References

Lesaffre, E. and Spiessens, B. (2001). On the effect of the number of quadrature points in a logistic random-effects model: An example. Journal of the Royal Statistical Society, Series C, 50, 325-335.

G. Fitzmaurice, N. Laird and J. Ware (2004) Applied Longitudinal Analysis, Wiley and Sons, New York, USA

uniond *Union Membership*

Description

This data set consider growth information on wages and union membership for 534 workers. The datafile contains observations on 11 variables sampled from the Current Population Survey of 1985. This data set demonstrates multiple regression, confounding, transformations, multicollinearity, categorical variables, ANOVA, pooled tests of significance, interactions and model building strategies.

Usage

```
data(uniond)
```

Format

A data frame with 534 observations on the following 11 variables.

`education` a numeric vector giving the number of years of education.

`south` a numeric vector gving an indicator variable for Southern Region (1=Person lives in South, 0=Person lives elsewhere).

`sex` a numeric vector giving an indicator variable for sex (1=Female, 0=Male).

`experience` a numeric vector giving the number of years of work experience.

`unionv` a numeric vector giving an indicator variable for union membership (1=Union member, 0=Not union member).

`wage` a numeric vector giving the Wage (dollars per hour).

`age` a numeric vector giving the Age in years.

`race` a numeric vector giving the race (1=Other, 2=Hispanic, 3=White).

`occupation` a numeric vector giving the occupational category (1=Management, 2=Sales, 3=Clerical, 4=Service, 5=Professional, 6=Other).

`sector` a numeric vector giving the Sector (0=Other, 1=Manufacturing, 2=Construction).

`marr` a numeric vector giving the Marital Status (0=Unmarried, 1=Married).

Details

The Current Population Survey (CPS) is used to supplement census information between census years. These data consist of a random sample of 534 persons from the CPS, with information on wages and other characteristics of the workers, including sex, number of years of education, years of work experience, occupational status, region of residence and union membership. We wish to determine (i) whether wages are related to these characteristics and (ii) whether there is a gender gap in wages. Based on residual plots, wages were log-transformed to stabilize the variance. Age and work experience were almost perfectly correlated (r=.98). Multiple regression of log wages against sex, age, years of education, work experience, union membership, southern residence, and occupational status showed that these covariates were related to wages (pooled F test, $p < .0001$).

The effect of age was not significant after controlling for experience. Standardized residual plots showed no patterns, except for one large outlier with lower wages than expected. This was a male, with 22 years of experience and 12 years of education, in a management position, who lived in the north and was not a union member. Removing this person from the analysis did not substantially change the results, so that the final model included the entire sample. Adjusting for all other variables in the model, females earned 81 the wages of males ($p < .0001$). Wages increased 41 additional years of education ($p < .0001$). They increased by 11 for every additional 10 years of experience ($p < .0001$). Union members were paid 23 paid 11 positions were paid most, and service and clerical positions were paid least (pooled F-test, $p < .0001$). Overall variance explained was R2 = .35. In summary, many factors describe the variations in wages: occupational status, years of experience, years of education, sex, union membership and region of residence. However, despite adjustment for all factors that were available, there still appeared to be a gender gap in wages. There is no readily available explanation for this gender gap.

Examples

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

Index

∗Topic **datasets**
bir, 7
calgb, 8
calgb.pred, 9
deterioration, 14
fleabeetles, 109
fractionation, 122
galaxy, 123
hiv, 132
igg, 133
indon, 134
nodal, 200
orings, 202
psychiatric, 221
rats, 255
rolling, 256
schoolgirls, 257
seizures, 258
sports, 259
toenail, 263
uniond, 264
∗Topic **models**
BDPdensity, 4
CSDPbinary, 10
DPbetabinom, 15
DPbinary, 18
DPcdensity, 23
DPdensity, 28
DPelicit, 34
DPglmm, 36
DPlmm, 40
DPMdencens, 44
DPmeta, 49
DPMglmm, 53
DPMlmm, 57
DPMmeta, 61
DPMolmm, 65
DPMrandom, 69
DPMrasch, 71
DPMraschpoisson, 76
DPmultmeta, 80
DPolmm, 84
DPpsBF, 88
DPrandom, 89
DPrasch, 91
DPraschpoisson, 95
DProc, 100
DPsurvint, 104
FPTbinary, 110
FPTrasch, 113
FPTraschpoisson, 118
HDPMcdensity, 124
HDPMdensity, 128
LDBDPdensity, 135
LDDPdensity, 140
LDDPrasch, 145
LDDPraschpoisson, 152
LDDProc, 158
LDDPsurvival, 165
LDDPtwopl, 171
LDPDdoublyint, 177
LDTFPdensity, 182
LDTFPglmm, 190
LDTFPsurvival, 196
Pbinary, 203
Plm, 206
predict.DPsurvint, 208
predict.HDPMcdensity, 211
predict.HDPMdensity, 214
ps, 216
PSgam, 217
PTdensity, 222
PTglmm, 226
PTlm, 231
PTlmm, 236
PTmeta, 240
PTolmm, 244
PTrandom, 248

PTsampler, 250
TDPdensity, 260
∗Topic **nonparametric**
BDPdensity, 4
CSDPbinary, 10
DPbetabinom, 15
DPbinary, 18
DPcaterpillar, 22
DPcdensity, 23
DPdensity, 28
DPelicit, 34
DPglmm, 36
DPMdencens, 44
DPmeta, 49
DPMglmm, 53
DPMmeta, 61
DPMolmm, 65
DPMrandom, 69
DPMrasch, 71
DPMraschpoisson, 76
DPmultmeta, 80
DPolmm, 84
DPpsBF, 88
DPrandom, 89
DPrasch, 91
DPraschpoisson, 95
DProc, 100
DPsurvint, 104
FPTbinary, 110
FPTrasch, 113
FPTraschpoisson, 118
HDPMcdensity, 124
HDPMdensity, 128
LDBDPdensity, 135
LDDPdensity, 140
LDDPrasch, 145
LDDPraschpoisson, 152
LDDProc, 158
LDDPsurvival, 165
LDDPtwopl, 171
LDPDdoublyint, 177
LDTFPdensity, 182
LDTFPglmm, 190
LDTFPsurvival, 196
Plm, 206
predict.HDPMcdensity, 211
predict.HDPMdensity, 214
ps, 216
PSgam, 217
PTdensity, 222
PTglmm, 226
PTlm, 231
PTlmm, 236
PTmeta, 240
PTolmm, 244
PTrandom, 248
PTsampler, 250
TDPdensity, 260
∗Topic **plot**
DPcaterpillar, 22
∗Topic **regression**
ps, 216
∗Topic **smooth**
ps, 216

BDPdensity, 4, *26*, *31*, *224*, *262*
bir, 7

calgb, 8
calgb.pred, 9
CSDPbinary, 10

deterioration, 14
DPbetabinom, 15
DPbinary, 18
DPcaterpillar, 22
DPcdensity, 23, *138*, *143*, *185*
DPdensity, *6*, *26*, 28, *47*, *100–102*, *224*, *262*
DPelicit, 34
DPglmm, 36, *43*, *51*, *56*, *60*, *64*, *68*, *87*, *230*, *238*, *242*, *247*
DPlmm, *39*, 40, *51*, *56*, *60*, *64*, *68*, *87*, *230*, *238*, *242*, *247*
DPMdencens, 44
DPmeta, 49, *64*, *82*
DPMglmm, *39*, *43*, *51*, 53, *60*, *64*, *68*, *87*, *230*, *238*, *242*, *247*
DPMlmm, *39*, *43*, *51*, *56*, 57, *68*, *87*, *230*, *238*, *242*, *247*
DPMmeta, *51*, 61, *242*
DPMolmm, *39*, *43*, *51*, *56*, *60*, *64*, 65, *87*, *230*, *238*, *242*, *247*
DPMrandom, *56*, *60*, *64*, *68*, 69
DPMrasch, 71, *148*
DPMraschpoisson, 76, *155*
DPmultmeta, 80

DPolmm, *39*, *43*, *51*, *56*, *60*, *64*, *68*, 84, *230*, *238*, *242*, *247*
DPpsBF, 88
DPrandom, *31*, *39*, *43*, *47*, *51*, *74*, *78*, *87*, 89, *94*, *98*, *116*, *120*, *148*, *155*
DPrasch, *74*, 91, *116*, *148*
DPraschpoisson, *78*, 95, *120*, *155*
DProc, 100
DPsurvint, 104, *210*

fleabeetles, 109
FPTbinary, 110
FPTrasch, *74*, *94*, 113, *148*
FPTraschpoisson, *78*, *98*, 118, *155*
fractionation, 122

galaxy, 123

HDPMcdensity, 124, *212*
HDPMdensity, 128, *214, 215*
hiv, 132

igg, 133
indon, 134

LDBDPdensity, 135
LDDPdensity, *138*, 140, *160, 161*, *168*, *180*, *185*
LDDPrasch, 145, *174*
LDDPraschpoisson, 152
LDDProc, 158
LDDPsurvival, 165, *180*, *198*
LDDPtwopl, 170
LDPDdoublyint, 177
LDTFPdensity, 182, *193*, *198*
LDTFPglmm, 190
LDTFPsurvival, *193*, 196

nodal, 200

orings, 202

Pbinary, 203
Plm, 206
predict.DPsurvint, *106*, 208
predict.HDPMcdensity, *126*, 211
predict.HDPMdensity, *131*, 214
ps, 216
PSgam, 217
psychiatric, 221
PTdensity, *6*, *26*, *31*, 222, *252*, *262*
PTglmm, *39*, *43*, *56*, *60*, *68*, *87*, 226, *230*, *238*, *247*
PTlm, 231
PTlmm, *39*, *43*, *56*, *60*, *68*, *87*, 235, *247*
PTmeta, 240
PTolmm, *39*, *43*, *56*, *60*, *68*, *87*, *230*, *238*, 244
PTrandom, *230*, *238*, *242*, *247*, 248
PTsampler, 250

rats, 255
rolling, 256

schoolgirls, 257
seizures, 258
sports, 259

TDPdensity, 260
toenail, 263

uniond, 264