Package 'conting'

July 2, 2014

Type Package

Title Bayesian analysis of contingency tables

Version 1.3

Date 2014-06-11

Author Antony M. Overstall

Maintainer Antony M. Overstall <antony@mcs.st-and.ac.uk>

Description Bayesian analysis of complete and incomplete contingency tables.

Depends R (>= 2.15.0)

Imports mvtnorm, BMS, gtools, tseries, coda

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2014-06-11 11:57:55

R topics documented:

`conting-package` *Bayesian Analysis of Complete and Incomplete Contingency Tables*

Description

Performs Bayesian analysis of complete and incomplete contingency tables incorporating model uncertainty using log-linear models. These analyses can be used to identify associations/interactions between categorical factors and to estimate unknown closed populations.

Details

Package:	conting
Type:	Package
Version:	1.3
Date:	2014-06-11
License:	GPL-2

For the Bayesian analysis of complete contingency tables the key function is `bcct` which uses MCMC methods to generate a sample from the joint posterior distribution of the model parameters and model indicator. Further MCMC iterations can be performed by using `bcctu`.

For the Bayesian analysis of incomplete contingency tables the key function is bict which uses MCMC methods to generate a sample from the joint posterior distribution of the model parameters, model indicator and the missing, and, possibly, censored cell entries. Further MCMC iterations can be performed by using bictu.

In both cases see Overstall & King (2014), and the references therein, for details on the statistical and computational methods, as well as detailed examples.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>

Maintainer: Antony M. Overstall <antony@mcs.st-and.ac.uk>

References

Overstall, A.M. & King, R. (2014) conting: An R package for Bayesian analysis of complete and incomplete contingency tables. *Journal of Statistical Software*, **58 (7)**, 1–27. http://www.jstatsoft.org/v58/i07/

Examples

```
set.seed(1)
## Set seed for reproducibility
data(AOH)
## Load AOH data
test1<-bcct(formula=y~(alc+hyp+obe)^3,data=AOH,n.sample=100,prior="UIP")
## Bayesian analysis of complete contingency table. Let the saturated model
## be the maximal model and do 100 iterations.

summary(test1)
## Summarise the result. Will get:
#Posterior summary statistics of log-linear parameters:
#            post_prob post_mean post_var lower_lim upper_lim
#(Intercept)         1  2.877924 0.002574   2.78778   2.97185
#alc1                1 -0.060274 0.008845  -0.27772   0.06655
#alc2                1 -0.049450 0.006940  -0.20157   0.11786
#alc3                1  0.073111 0.005673  -0.05929   0.20185
#hyp1                1 -0.544988 0.003485  -0.65004  -0.42620
#obe1                1 -0.054672 0.007812  -0.19623   0.12031
#obe2                1  0.007809 0.004127  -0.11024   0.11783
#NB: lower_lim and upper_lim refer to the lower and upper values of the
#95 % highest posterior density intervals, respectively
#
#Posterior model probabilities:
#  prob model_formula
#1 0.45 ~alc + hyp + obe
#2 0.30 ~alc + hyp + obe + hyp:obe
#3 0.11 ~alc + hyp + obe + alc:hyp + hyp:obe
#4 0.06 ~alc + hyp + obe + alc:hyp + alc:obe + hyp:obe
#5 0.05 ~alc + hyp + obe + alc:hyp
#
#Total number of models visited =  7
```

```
#
#Under the X2 statistic
#
#Summary statistics for T_pred
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  11.79   20.16   23.98   24.70   28.77   52.40
#
#Summary statistics for T_obs
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#   8.18   24.22   31.51   30.12   35.63   42.49
#
#Bayesian p-value =  0.28

set.seed(1)
## Set seed for reproducibility
data(spina)
## Load spina data
test2<-bict(formula=y~(S1+S2+S3+eth)^2,data=spina,n.sample=100,prior="UIP")
## Bayesian analysis of incomplete contingency table. Let the model with two-way
## interactions be the maximal model and do 100 iterations.

summary(test2)
## Summarise the result. Will get:

#Posterior summary statistics of log-linear parameters:
#            post_prob post_mean post_var lower_lim upper_lim
#(Intercept)         1    1.0427 0.033967    0.6498    1.4213
#S11                 1   -0.3159 0.015785   -0.4477   -0.1203
#S21                 1    0.8030 0.018797    0.6127    1.1865
#S31                 1    0.7951 0.003890    0.6703    0.8818
#eth1                1    2.8502 0.033455    2.4075    3.1764
#eth2                1    0.1435 0.072437   -0.4084    0.5048
#S21:S31             1   -0.4725 0.002416   -0.5555   -0.3928
#NB: lower_lim and upper_lim refer to the lower and upper values of the
#95 % highest posterior density intervals, respectively
#
#Posterior model probabilities:
#  prob model_formula
#1 0.36 ~S1 + S2 + S3 + eth + S2:S3
#2 0.19 ~S1 + S2 + S3 + eth + S2:S3 + S2:eth
#3 0.12 ~S1 + S2 + S3 + eth + S1:eth + S2:S3
#4 0.12 ~S1 + S2 + S3 + eth + S1:S2 + S1:S3 + S1:eth + S2:S3 + S2:eth + S3:eth
#5 0.10 ~S1 + S2 + S3 + eth + S1:S3 + S1:eth + S2:S3
#6 0.06 ~S1 + S2 + S3 + eth + S1:S3 + S1:eth + S2:S3 + S2:eth
#Total number of models visited =  8
#
#Posterior mean of total population size = 726.75
#95 % highest posterior density interval for total population size = ( 706 758 )
#
#Under the X2 statistic
#
#Summary statistics for T_pred
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
#  8.329  15.190  20.040  22.550  24.180 105.200
#
#Summary statistics for T_obs
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# 5.329  18.270  22.580  21.290  24.110  37.940
#
#Bayesian p-value =  0.45
```

`accept_rate` *Compute Acceptance Rates for Metropolis-Hastings and Reversible Jump Algorithms*

Description

This function computes the acceptance rates of the Metropolis-Hastings and reversible jump algorithms from the MCMC output of `bcct` and `bict` objects.

Usage

```
accept_rate(object)
```

Arguments

`object`	An object of class `"bcct"` or `"bict"`.

Details

Acceptance rates can be used to assess the performance of MCMC methods (in particular the peformance of the reversible jump method, Brooks et al, 2003).

Value

This function will return an object of class `"acceptrate"` which is a list with the following components.

`rj_ar`	Acceptance rate (as a %) of the reversible jump algorithm.
`mh_ar`	Acceptance rate (as a %) of the Metropolis-Hastings algorithm.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

References

Brooks, S.P., Giudici, P., & Roberts, G.O. (2003) Efficient construction of reversible jump Markov chain Monte Carlo proposal distributions. *Journal of the Royal Statistical Society, Series B*, **65 (1)**, 3–55.

See Also

print.acceptrate, bcct, bict.

Examples

```
set.seed(1) ## set a seed for reproducibility
data(AOH)
test1<-bcct(formula=y~(alc+hyp+obe)^3,data=AOH,n.sample=500,prior="UIP")
## Create a bcct object for the AOH dataset for a very small number of
## iterations  (500).
accept_rate(test1)
## Calculate accept rates. Will get:

#Acceptance rate of reversible jump proposals =  32.5581 %
#Acceptance rate of Metropolis-Hastings proposals =  76.8595 %
```

---

add_term *Determines Model Moves Given Current Model*

Description

These functions are used to detemine which models we can propose moves to, given the current model in the MCMC algorithm, and the principle of marginality.

Usage

```
add_term(curr.index, data, maximal.mod)
drop_term(curr.index, data, maximal.mod)
prop_mod(curr.index,data,maximal.mod,null.move.prob=0.5)
```

Arguments

curr.index	A binary vector, of the same length as the number of log-linear parameters in the maximal model, indicating which parameters are present in the current model.
data	An object of class "data.frame" containing the variables in the model.
maximal.mod	An object of class "glm" giving the fit of the maximal model.
null.move.prob	An optional scalar argument giving the probability of performing a null move, i.e. proposing a move to the current model. The default value is 0.5.

Details

In the reversible jump algorithm we propose a move to a model given the current model. The function `prop_mod` implements a scheme whereby only local proposals are made, i.e. either a term is added or dropped. These types of move are called birth and death moves, respectively, by Forster et al (2012).

When a term is either added or dropped, we preserve the principle of marginality, e.g. we can only propose to add a three-way interaction if all the possible two-way interactions between the three factors are included in the present model.

The functions `add_term` and `drop_term` determine which terms can be added or dropped whilst preserving the principle of marginality.

The function `prop_mod` will call `add_term` and `drop_term` thus determining which terms can be added or dropped. With probability `null.move.prob` it will choose to remain in the current model; otherwise it will choose one of the possible terms to add or drop.

Value

The functions `add_term` and `drop_term` will output a character vector containing the names of terms that can be dropped.

The function `prop_mod` will return a list with the following components.

`new.index`	A binary vector, of the same length as the number of log-linear parameters in the maximal model, indicating which parameters are present in the proposed model.
`type`	A character string which will be one of `c("null","drop","add")` depending on the type of move proposed.
`total.choices`	If `type` is not `"null"`, then `total.choices` will be scalar giving the total number of non-null moves available. If `type` is equal to `"null"`, then `total.choices` will be 0.
`null.move.prob`	A scalar giving the probability of a null move.

Note

This function will not typically be called by the user.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

References

Forster, J.J., Gill, R.C. & Overstall, A.M. (2012) Reversible jump methods for generalised linear models and generalised linear mixed models. *Statistics and Computing*, **22 (1)**, 107–120.

Examples

```
data(AOH)
## Load the AOH data

maximal.mod<-glm(formula=y~(alc+hyp+obe)^3,data=AOH,x=TRUE,y=TRUE,
contrasts=list(alc="contr.sum",hyp="contr.sum",obe="contr.sum"))
## Set up the maximal model which in this case is the saturated model.

curr.index<-formula2index(big.X=maximal.mod$x,formula=y~alc+hyp+obe+hyp:obe,data=AOH)
## Set up the binary vector for the model containing all main effects and the
## hyp:obe interaction.

add_term(curr.index=curr.index,data=AOH,maximal.mod=maximal.mod)
## See what terms we can add - will get:

#[1] "alc:hyp" "alc:obe"

drop_term(curr.index=curr.index,data=AOH,maximal.mod=maximal.mod)
## See what terms we can drop - will get:

#[1] "hyp:obe"

set.seed(4)
## Set the seed for reproducibility.

prop_mod(curr.index=curr.index,data=AOH,maximal.mod=maximal.mod)
## Propose a model. Will be a drop move, proposing the independence model by
## dropping the hyp:obe interaction. The total.choices object is 3, i.e. one
## drop move and two add moves. Specifically:

#$new.index
# [1] 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
#
#$type
#[1] "drop"
#
#$total.choices
#[1] 3
#
#$null.move.prob
#[1] 0.5
```

AOH *Alcohol, Obesity and Hypertension: A Complete 4 * 3 * 2 Table*

Description

491 subjects are cross-classified according to the three factors: hypertension (hyp; 2 levels), obesity (obe; 3 levels) and alcohol (alc; 4 levels). There are a total of 24 cells in the table.

Usage

```
data(AOH)
```

Format

A "data.frame" with 24 observations on the following 4 variables.

`y` Counts in each cell of table.

`alc` A factor with levels `0` `1-2` `3-5` `6+` indicating the classification of alcohol intake of drinks per day.

`obe` A factor with levels `low` `average` `high` indicating the classification of obesity.

`hyp` A factor with levels `yes` `no` indicating the classification of hypertension.

Details

These data are from a study in Western Australia. The study copied a larger study from USA. See Knuiman & Speed (1988) for more details.

For details on the function `bcct` applied to these data, see Overstall & King (2014).

Source

Knuiman, M.W. & Speed, T.P. (1988) Incorporating Prior Information into the Analysis of Contingency Tables. *Biometrics*, **44 (4)**, 1061–1071.

References

Overstall, A.M. & King, R. (2014) conting: An R package for Bayesian analysis of complete and incomplete contingency tables. *Journal of Statistical Software*, **58 (7)**, 1–27. http://www.jstatsoft.org/v58/i07/

Examples

```
data(AOH)
summary(AOH)
```

---

bayespval *Compute Bayesian p-value*

Description

This function will compute the Bayesian (or posterior predictive) p-value. This can be used as a diagnostic tool to check model adequacy. Additionally this function outputs predictions from the model which can also be used in other assessments of model adequacy.

Usage

```
bayespval(object, n.burnin = 0, thin = 1, statistic = "X2")
```

Arguments

object	An object of class "bcct" or "bict" object.
n.burnin	An optional argument giving the number of iterations to use as burn-in. The default value is 0.
thin	An optional argument giving the amount of thinning to use, i.e. the computations are based on every thin-th value in the MCMC sample. The default value is 1, i.e. no thinning.
statistic	An optional argument giving the discrepancy statistic to use for calculating the Bayesian p-value. It can be one of c("X2","FreemanTukey","deviance") which correspond to the different statistics: "X2" = Chi-squared statistic, "FreemanTukey" = Freeman-Tukey statistic, "deviance" = deviance statistic. See Overstall & King (2014), and references therein, for descriptions of these statistics.

Details

See Gelman et al (2004, Chapter 6) for more details on Bayesian p-values and see Overstall & King (2014), and references therein, for details of their application to contingency tables.

The use of thinning is recommended when the number of MCMC iterations and/or the number of log-linear parameters in the maximal model are/is large, which may cause problems with comuter memory storage.

Value

The function will produce an object of class "pval" which is a list with the following components.

PRED	An (n.sample-n.burnin) by n* matrix where (n* is the number of observed cell counts) containing the predictions of the observed cell counts.
Tpred	A vector of length (n.sample-n.burnin) containing the discrepancies between the predicted cell counts and their means.
Tobs	A vector of length (n.sample-n.burnin) containing the discrepancies between the observed cell counts and their means.
pval	A scalar giving the Bayesian p-value, i.e. the proportion of Tpred>Tobs.
statnum	A numeric scalar identifying which statistic is used.
statistic	A character string identifying which statistic is used.
thin	The value of the argument thin.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

References

Gelman, A., Carlin, J.B., Stern, H.S. & Rubin, D.B. (2004) *Bayesian Data Analysis*, 2nd edition, Chapman & Hall.

Overstall, A.M. & King, R. (2014) conting: An R package for Bayesian analysis of complete and incomplete contingency tables. *Journal of Statistical Software*, **58 (7)**, 1–27. http://www.jstatsoft.org/v58/i07/

See Also

bict, bcct, print.pval.

Examples

```
set.seed(1)
## Set seed for reproducibility
data(spina)
## Load spina data

test1<-bict(formula=y~(S1+S2+S3+eth)^2,data=spina,n.sample=50,prior="UIP")
## Do 50 iterations starting at maximal model containing all two-way interactions.

test1p<-bayespval(object=test1,statistic="FreemanTukey",n.burnin=5)
## Use the Freeman-Tukey statistic and a burn-in phase of 5 iterations.
test1p
## Will get following output

#Under the Freeman-Tukey statistic
#
#Summary statistics for T_pred
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  2.812   4.695   5.190   5.777   6.405  14.490
#
#Summary statistics for T_obs
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  4.566   4.861   5.197   5.430   6.108   6.460
#
#Bayesian p-value =  0.4667

## Can do a plot

## Not run: plot(test1p)
```

---

bcct *Bayesian Analysis of Complete Contingency Tables*

Description

These functions implement a Bayesian analysis of complete contingency tables. This is accomplished using an MCMC algorithm where the null moves are performed using a Metropolis-Hastings algorithm and the between models moves are performed using a reversible jump algorithm.

bcct should be used initially, and bcctu should be used to do additional MCMC iterations, if required.

Usage

```
bcct(formula, data, n.sample, prior = "SBH", start.formula = NULL,
start.beta = NULL, start.sig = NULL, save = 0, name = NULL, null.move.prob=0.5,
a = 0.001, b = 0.001, progress = FALSE)

bcctu(object, n.sample, save = NULL, name = NULL, progress = FALSE)
```

Arguments

formula	An object of class "formula": a symbolic description of the maximal model.
object	An object of class "bcct" produced as a previous call to bcct or bcctu.
data	An object of class "data.frame" (or "table") containing the variables in the model. If the model variables are not found in data, the variables are taken from environment(formula), typically the environment from which bcct is called.
n.sample	A numeric scalar giving the number of (additional, in the case of bcctu) MCMC iterations to perform.
prior	An optional argument giving the prior to be used in the analysis. It can be one of c("UIP","SBH"), where "UIP" = unit information prior; and "SBH" = Sabanes-Bove & Held prior. The default value is "SBH".
start.formula	An optional argument giving an object of class "formula": a symbolic description of the starting model in the MCMC algorithm. If NULL (the default) the starting model will be the maximal model.
start.beta	An optional argument giving the starting values of the log-linear parameters for the MCMC algorithm. It should be a vector of the same length as the number of log-linear parameters in the starting model implied by the argument start.formula. If NULL (the default) the starting value will be the posterior mode under the maximal model.
start.sig	An optional argument giving the starting value of sigma^2 (under the Sabanes-Bove & Held prior) for the MCMC algorithm when the argument of prior is "SBH". If NULL (the default) the starting value will be one.
save	An optional argument for saving the MCMC output mid-algorithm. For bcct, if positive, the function will save the MCMC output to external text files every save iterations. If zero (the default), the function will not save the MCMC output to external files. For bcctu, if non-NULL, the function will save the MCMC output to external text files every save iterations. If NULL (the default), it will inherit the value of save from the previous call to bcct or bcctu.
name	An optional argument giving a prefix to the file name of the external files saved if the argument save is positive. For bcct, a value of NULL means the external files will not have a prefix. For bcctu, a value of NULL, means the prefix will be inherited from the previous call to bcct or bcctu.
null.move.prob	An optional scalar argument giving the probability of performing a null move in the reversible jump algorithm, i.e. proposing a move to the current model. The default value is 0.5.

a	The shape hyperparameter of the Sabanes-Bove & Held prior, see Overstall & King (2014). The default value is 0.001.
b	The scale hyperparameter of the Sabanes-Bove & Held prior, see Overstall & King (2014). The default value is 0.001.
progress	Logical argument. If TRUE, then a progress bar will be displayed. The default value is FALSE.

Details

For identifiability, the parameters are constrained. The conting-package uses sum-to-zero constraints. See Overstall & King (2014), and the references therein, for more details.

The Metropolis-Hastings algorithm employed is the iterated weighted least squares method for generalised linear models (GLMs) proposed by Gamerman (1997). The reversible jump algorithm employed is that orthogonal projections method for GLMs proposed by Forster et al (2012). For details on these methods applied to log-linear models see Overstall & King (2014), and the references therein.

For details on the unit information and Sabanes-Bove & Held priors for generalised linear models see Ntzoufras et al (2003) and Sabanes-Bove & Held (2011), respectively. See Overstall & King (2014), and the references therein, for their application to log-linear models and contingency tables.

Value

The functions will return an object of class "bcct" which is a list with the following components:

BETA	An n.sample by p matrix containing the sampled values of the log-linear parameters, where p is the number of log-linear parameters in the maximal model. For elements of this matrix which correspond to a log-linear parameter which is not present for the current model a zero is returned.
MODEL	A vector of length n.sample giving the sampled model indicators in hexadecimal format.
SIG	A vector of length n.sample giving the sampled values for sigma^2 under the Sabanes-Bove & Held prior. If the unit information prior is used then the components of this vector will be one.
rj_acc	A binary vector of the same length as the number of reversible jump moves attempted. A 0 indicates that the proposal was rejected, and a 1 that the proposal was accepted.
mh_acc	A binary vector of the same length as the number of Metropolis-Hastings moves attempted. A 0 indicates that the proposal was rejected, and a 1 that the proposal was accepted.
priornum	A numeric scalar indicating which prior was used: 1 = "UIP", 2 = "SBH".
maximal.mod	An object of class "glm" giving the fit of the maximal model.
IP	A p by p matrix giving the inverse of the prior scale matrix for the maximal model.
eta.hat	A vector of length n (number of cells) giving the posterior mode of the linear predictor under the maximal model.

save	The argument save.
name	The argument name.
null.move.prob	The argument null.move.prob.
time	The total computer time (in seconds) used for the MCMC computations.
a	The argument a.
b	The argument b.

Note

These functions are wrappers for bcct.fit.

In Version 1.0 of conting-package, note that the default value for prior was "UIP". From Version 1.1 onwards, the default value is "SBH".

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

References

Sabanes-Bove, D. & Held, L. (2011) Hyper-g priors for generalized linear models. *Bayesian Analysis*, **6 (3)**, 387–410.

Forster, J.J., Gill, R.C. & Overstall, A.M. (2012) Reversible jump methods for generalised linear models and generalised linear mixed models. *Statistics and Computing*, **22 (1)**, 107–120.

Gamerman, D. (1997) Sampling from the posterior distribution in generalised linear mixed models. *Statistics and Computing*, **7 (1)**, 57–68.

Nztoufras, I., Dellaportas, P. & Forster, J.J. (2003) Bayesian variable and link determination for generalised linear models. *Journal of Statistical Planning and Inference*, **111 (1)**, 165–180.

Overstall, A.M. & King, R. (2014) conting: An R package for Bayesian analysis of complete and incomplete contingency tables. *Journal of Statistical Software*, **58 (7)**, 1–27. http://www.jstatsoft.org/v58/i07/

See Also

bcct.fit, AOH, heart.

Examples

```
set.seed(1)
## Set seed for reproducibility.

data(AOH)
## Load the AOH data

test1<-bcct(formula=y~(alc+hyp+obe)^3,data=AOH,n.sample=50,prior="UIP")
## Let the maximal model be the saturated model. Starting from the
## posterior mode of the maximal model do 50 iterations under the unit
## information prior.
```

```
test1<-bcctu(object=test1,n.sample=50)
## Do another 50 iterations

test1
## Printing out a bcct object produces this simple summary

#Number of cells in table = 24
#
#Maximal model =
#y ~ (alc + hyp + obe)^3
#
#Number of log-linear parameters in maximal model = 24
#
#Number of MCMC iterations = 100
#
#Computer time for MCMC = 00:00:01
#
#Prior distribution for log-linear parameters = UIP

summary(test1)
## Printing out a summary produces a bit more:

#Posterior summary statistics of log-linear parameters:
#            post_prob post_mean post_var lower_lim upper_lim
#(Intercept)         1  2.877924 0.002574   2.78778   2.97185
#alc1                1 -0.060274 0.008845  -0.27772   0.06655
#alc2                1 -0.049450 0.006940  -0.20157   0.11786
#alc3                1  0.073111 0.005673  -0.05929   0.20185
#hyp1                1 -0.544988 0.003485  -0.65004  -0.42620
#obe1                1 -0.054672 0.007812  -0.19623   0.12031
#obe2                1  0.007809 0.004127  -0.11024   0.11783
#NB: lower_lim and upper_lim refer to the lower and upper values of the
#95 % highest posterior density intervals, respectively
#
#Posterior model probabilities:
#  prob model_formula
#1 0.45 ~alc + hyp + obe
#2 0.30 ~alc + hyp + obe + hyp:obe
#3 0.11 ~alc + hyp + obe + alc:hyp + hyp:obe
#4 0.06 ~alc + hyp + obe + alc:hyp + alc:obe + hyp:obe
#5 0.05 ~alc + hyp + obe + alc:hyp
#
#Total number of models visited =  7
#
#Under the X2 statistic
#
#Summary statistics for T_pred
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  11.79   20.16   23.98   24.70   28.77   52.40
#
#Summary statistics for T_obs
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
#   8.18   24.22   31.51   30.12   35.63   42.49
#
#Bayesian p-value =  0.28

## For more examples see Overstall & King (2014).
```

---

bcct.fit *Bayesian Analysis of Complete Contingency Tables*

Description

This function is the workhorse behind bcct and bcctu.

Usage

```
bcct.fit(priornum, maximal.mod, IP, eta.hat, ini.index, ini.beta, ini.sig,
iters, save, name, null.move.prob, a, b, progress)
```

Arguments

priornum	A numeric scalar indicating which prior is to be used: 1 = "UIP", 2 = "SBH".
maximal.mod	An object of class "glm" giving the fit of the maximal model.
IP	A p by p matrix giving the inverse of the prior scale matrix for the maximal model.
eta.hat	A vector of length n (number of cells) giving the posterior mode of the linear predictor under the maximal model.
ini.index	A binary vector, of the same length as the number of log-linear parameters in the maximal model, indicating which parameters are present in the initial model.
ini.beta	A numeric vector giving the starting values of the log-linear parameters for the MCMC algorithm.
ini.sig	A numeric scalar giving the starting value of sigma^2 for the MCMC algorithm.
iters	The number of iterations of the MCMC algorithm to peform.
save	If positive, the function will save the MCMC output to external text files every save iterations. If zero , the function will not save the MCMC output to external files.
name	A prefix to the external files saved if the argument save is positive. If NULL, then the external files will have no prefix.
null.move.prob	A scalar argument giving the probability of performing a null move, i.e. proposing a move to the current model.
a	The shape hyperparameter of the Sabanes-Bove & Held prior, see Overstall & King (2014).
b	The scale hyperparameter of the Sabanes-Bove & Held prior, see Overstall & King (2014).
progress	Logical argument. If TRUE, then a progress bar will be displayed.

Value

The function will return a list with the following components:

BETA	An iters by p matrix containing the sampled values of the log-linear parameters, where p is the number of log-linear parameters in the maximal model. For elements of this matrix which correspond to a log-linear parameter which is not present for the current model a zero is returned.
MODEL	A vector of length iters giving the sampled model indicators in hexadecimal form.
SIG	A vector of length iters giving the sampled values for sigma^2 under the Sabanes-Bove & Held prior. If the unit information prior is used then the components of this vector will be one.
rj_acc	A binary vector of the same length as the number of reversible jump moves attempted. A 0 indicates that the proposal was rejected, and a 1 that the proposal was accepted.
mh_acc	A binary vector of the same length as the number of Metropolis-Hastings moves attempted. A 0 indicates that the proposal was rejected, and a 1 that the proposal was accepted.

Note

This function will not typically be called by the user.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

References

Overstall, A.M. & King, R. (2014) conting: An R package for Bayesian analysis of complete and incomplete contingency tables. *Journal of Statistical Software*, **58 (7)**, 1–27. http://www.jstatsoft.org/v58/i07/

See Also

bcct, bcctu.

Examples

```
data(AOH)
## Load the AOH data.

maximal.mod<-glm(formula=y~(alc+hyp+obe)^3,data=AOH,x=TRUE,y=TRUE,
contrasts=list(alc="contr.sum",hyp="contr.sum",obe="contr.sum"))
## Set up the maximal model which in this case is the saturated
## model.

curr.index<-formula2index(big.X=maximal.mod$x,formula=y~alc+hyp+obe+hyp:obe,data=AOH)
## Set up the binary vector for the model containing all main effects and the
```

```
## hyp:obe interaction.

IP<-t(maximal.mod$x)%*%maximal.mod$x/length(maximal.mod$y)
IP[,1]<-0
IP[1,]<-0
## Set up the inverse scale matrix for the prior distribution under
## the maximal model.

bmod<-beta_mode(X=maximal.mod$x,prior="UIP",y=maximal.mod$y,IP=IP)
## Find the posterior mode under the maximal model

eta.hat<-as.vector(maximal.mod$x%*%bmod)
## Find the posterior mode of the linear predictor
## under the maximal model.

set.seed(1)
## Set seed for reproducibility

test1<-bcct.fit(priornum=1, maximal.mod=maximal.mod, IP=IP, eta.hat=eta.hat,
ini.index=curr.index, ini.beta=bmod[curr.index==1], ini.sig=1, iters=5, save=0,
name=NULL,null.move.prob=0.5, a=0.001, b=0.001, progress=TRUE)
## Run for 5 iterations starting at model defined by curr.index.

test1$MODEL
## Look at sampled model indicators. Should be:
## [1] "fe00c0" "fe0000" "fe0000" "fe0000" "fe0000"

model2index(test1$MODEL,dig=24)
## Convert these to binary indicators of the log-linear parameters.
## Will get:

#       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
#fe00c0    1    1    1    1    1    1    1    0    0     0     0     0     0
#fe0000    1    1    1    1    1    1    1    0    0     0     0     0     0
#fe0000    1    1    1    1    1    1    1    0    0     0     0     0     0
#fe0000    1    1    1    1    1    1    1    0    0     0     0     0     0
#fe0000    1    1    1    1    1    1    1    0    0     0     0     0     0
#       [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
#fe00c0     0     0     0     1     1     0     0     0     0     0     0
#fe0000     0     0     0     0     0     0     0     0     0     0     0
#fe0000     0     0     0     0     0     0     0     0     0     0     0
#fe0000     0     0     0     0     0     0     0     0     0     0     0
#fe0000     0     0     0     0     0     0     0     0     0     0     0

## See how the hyp:obe interactions in columns 17 and 18 gets dropped after
## the 1st iteration.
```

---

beta_mode *Posterior Mode*

Description

This function finds the posterior mode of the log-linear parameters of a log-linear model with a given design matrix and prior distribution.

Usage

```
beta_mode(X, prior = "SBH", y, IP , a = 0.001 , b = 0.001)
```

Arguments

X	The n by p design matrix where n is the number of cells and p is the number of log-linear parameters.
prior	The prior distribution. It can be one of c("UIP","SBH"), where "UIP" = unit information prior; and "SBH" = Sabanes-Bove & Held prior. The default value is "SBH".
y	The n by 1 vector of cell counts.
IP	A p by p matrix giving the inverse of the prior scale matrix.
a	The shape hyperparameter of the Sabanes-Bove & Held prior, see Overstall & King (2014).
b	The scale hyperparameter of the Sabanes-Bove & Held prior, see Overstall & King (2014).

Details

The posterior mode is found by maximising the log unnormalised posterior pdf given by the sum of the log-likelihood and the log of the prior pdf. This optimisation is achieved using a quasi Newton-Raphson method.

For details on the unit information and Sabanes-Bove & Held priors for generalised linear models see Ntzoufras et al (2003) and Sabanes-Bove & Held (2011), respectively. See Overstall & King (2014), and the references therein, for their application to log-linear models and contingency tables.

The posterior mode is required for the reversible jump algorithm implemented from Forster et al (2012).

Value

beta_mode will return a p by 1 vector containing the posterior mode of the log linear parameters.

Note

This function will not typically be called by the user.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

References

Sabanes-Bove, D. & Held, L. (2011) Hyper-g priors for generalized linear models. *Bayesian Analysis*, **6 (3)**, 387–410.

Forster, J.J., Gill, R.C. & Overstall, A.M. (2012) Reversible jump methods for generalised linear models and generalised linear mixed models. *Statistics and Computing*, **22 (1)**, 107–120.

Nztoufras, I., Dellaportas, P. & Forster, J.J. (2003) Bayesian variable and link determination for generalised linear models. *Journal of Statistical Planning and Inference*, **111 (1)**, 165–180.

Overstall, A.M. & King, R. (2014) conting: An R package for Bayesian analysis of complete and incomplete contingency tables. *Journal of Statistical Software*, **58 (7)**, 1–27. http://www.jstatsoft.org/v58/i07/

Examples

```
data(AOH) ## loads the AOH data

X<-model.matrix(~alc+hyp+obe,data=AOH,
contrasts=list(alc="contr.sum",hyp="contr.sum",obe="contr.sum"))
## Sets up the design matrix for the independence model

IP<-(t(X)%*%X)/dim(X)[1]
## Set up inverse of prior scale matrix

beta_mode(X=X,prior="UIP",y=AOH$y,IP=IP)
## Finds the posterior mode of the log-linear parameters under the
## independence model with the unit information prior. Will get:
#X(Intercept)        Xalc1        Xalc2        Xalc3        Xhyp1        Xobe1
# 2.894270420 -0.045859743 -0.071775824  0.089541068 -0.504141954  0.008163604
#       Xobe2
#-0.016327209

beta_mode(X=X,prior="SBH",y=AOH$y,IP=IP)
## Finds the posterior mode of the log-linear parameters under the
## independence model with the Sabanes-Bove & Held prior. Will get:
#X(Intercept)        Xalc1        Xalc2        Xalc3        Xhyp1        Xobe1
# 2.908298763 -0.043704371 -0.068212247  0.085338704 -0.473628107  0.007762839
#       Xobe2
#-0.015525678
```

`bict` *Bayesian Analysis of Incomplete Contingency Tables*

Description

These functions implement a Bayesian analysis of incomplete contingency tables. This is accomplished using a data augmentation MCMC algorithm where the null moves are performed using the Metropolis-Hastings algorithm and the between models moves are performed using the reversible

jump algorithm. This function can also accomodate cases where one of the sources observes a mixture of individuals from target and non-target populations. This results in the some of the cell counts being censored.

`bict` should be used initially, and `bictu` should be used to do additional MCMC iterations, if needed.

Usage

```
bict(formula, data, n.sample, prior = "SBH", cens = NULL, start.formula = NULL,
start.beta = NULL, start.sig = NULL, start.y0 = NULL, save = 0, name = NULL,
null.move.prob=0.5, a = 0.001, b = 0.001, progress = FALSE)

bictu(object, n.sample, save = NULL, name = NULL, progress = FALSE)
```

Arguments

`formula`	An object of class `"formula"`: a symbolic description of the maximal model.
`object`	An object of class `"bict"` produced as a previous call to `bict` or `bictu`.
`data`	An object of class `"data.frame"` (or `"table"`) containing the variables in the model. If the model variables are not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `bict` is called.
`n.sample`	A numeric scalar giving the number of MCMC iterations to peform.
`prior`	An optional argument giving the prior to be used in the analysis. It can be one of `c("UIP","SBH")`, where `"UIP"` = unit information prior; and `"SBH"` = Sabanes-Bove & Held prior. The default value is `"SBH"`.
`cens`	A numeric vector indicating the row numbers of the data.frame in `data` which correspond to the censored cells. This can be found using the function `find_cens`.
`start.formula`	An optional argument giving an object of class `"formula"`: a symbolic description of the starting model in the MCMC algorithm. If `NULL` (the default) the starting model will be the maximal model.
`start.beta`	An optional argument giving the starting values of the log-linear parameters for the MCMC algorithm. It should be a vector of the same length as the number of log-linear parameters in the starting model implied by the argument `start.formula`. If `NULL` (the default) the starting value will be the posterior mode under the maximal model.
`start.sig`	An optional argument giving the starting value of sigma^2 (under the Sabanes-Bove & Held prior) for the MCMC algorithm when the argument of prior is `"SBH"`. If `NULL` (the default) the starting value will be one.
`start.y0`	An optional argument giving the starting values of the missing and censored cell counts. This should have the same length as the number of missing and censored cell counts.
`save`	An optional argument for saving the MCMC output mid-algorithm. For `bict`, if positive, the function will save the MCMC output to external text files every `save` iterations. If zero (the default), the function will not save the MCMC output to external files.

	For bictu, if non-NULL, the function will save the MCMC output to external text files every save iterations. If NULL (the default), it will inherit the value of save from the previous call to bict or bictu.
name	An optional argument giving a prefix to the external files saved if the argument save is positive. For bict, a value of NULL means the external files will not have a prefix. For bictu, a value of NULL, means the prefix will be inherited from the previous call to bict or bictu.
null.move.prob	An optional scalar argument giving the probability of performing a null move in the reversible jump algorithm, i.e. proposing a move to the current model. The default value is 0.5.
a	The shape hyperparameter of the Sabanes-Bove & Held prior, see Overstall & King (2014). The default value is 0.001.
b	The scale hyperparameter of the Sabanes-Bove & Held prior, see Overstall & King (2014). The default value is 0.001.
progress	Logical argument. If TRUE, then a progress bar will be displayed. The default value is FALSE.

Details

For identifiability, the parameters are constrained. The conting-package uses sum-to-zero constraints. See Overstall & King (2014), and the references therein, for more details.

The Metropolis-Hastings algorithm employed is the iterated weighted least squares method for generalised linear models (GLMs) proposed by Gamerman (1997). The reversible jump algorithm employed is the orthogonal projections method for GLMs proposed by Forster et al (2012). For details on these methods applied to log-linear models through the data-augmentation algorithm see Overstall & King (2014), and the references therein. For details on the censored approach see Overstall et al (2014).

For details on the unit information and Sabanes-Bove & Held priors for generalised linear models see Ntzoufras et al (2003) and Sabanes-Bove & Held (2011), respectively. See Overstall & King (2014), and the references therein, for their application to log-linear models and contingency tables.

Value

The functions will return an object of class "bict" which is a list with the following components.

BETA	An n.sample by p matrix containing the sampled values of the log-linear parameters, where p is the number of log-linear parameters in the maximal model. For elements of this matrix which correspond to a log-linear parameter which is not present for the current model a zero is returned.
MODEL	A vector of length n.sample giving the sampled model indicators in hexadecimal format.
SIG	A vector of length n.sample giving the sampled values for sigma^2 under the Sabanes-Bove & Held prior. If the unit information prior is used then the components of this vector will be one.
Y0	An n.sample by k matrix giving the sampled values of the missing and censored cell counts, where k is the total number of missing and censored cell counts.

`missing1` A vector of the same length as the number of missing cell counts giving the row numbers of the `data.frame` in `data` (or the elements of the variables) which correspond to the missing cell counts.

`missing2` A vector of the same length as the number of censored cell counts giving the row numbers of the `data.frame` in `data` (or the elements of the variables) which correspond to the censored cell counts.

`missing_details` The rows of the `data.frame` in `data` (or the elements of the variables) which correspond to the missing cell counts.

`censored_details` The rows of the `data.frame` in `data` (or the elements of the variables) which correspond to the censored cell counts.

`rj_acc` A binary vector of the same length as the number of reversible jump moves attempted. A 0 indicates that the proposal was rejected, and a 1 that the proposal was accepted.

`mh_acc` A binary vector of the same length as the number of Metropolis-Hastings moves attempted. A 0 indicates that the proposal was rejected, and a 1 that the proposal was accepted.

`priornum` A numeric scalar indicating which prior was used: 1 = `"UIP"`, 2 = `"SBH"`.

`maximal.mod` An object of class `"glm"` giving the fit of the maximal model.

`IP` A p by p matrix giving the inverse of the prior scale matrix for the maximal model.

`eta.hat` A vector of length n (number of cells) giving the posterior mode of the linear predictor under the maximal model.

`save` The argument `save`.

`name` The argument `name`.

`null.move.prob` The argument `null.move.prob`.

`time` The total computer time (in seconds) used for the MCMC computations.

`a` The argument `a`.

`b` The argument `b`.

Note

These functions are wrappers for `bict.fit`.

In Version 1.0 of `conting-package`, note that the default value for `prior` was `"UIP"`. From Version 1.1 onwards, the default value is `"SBH"`.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

References

Sabanes-Bove, D. & Held, L. (2011) Hyper-g priors for generalized linear models. *Bayesian Analysis*, **6 (3)**, 387–410.

Forster, J.J., Gill, R.C. & Overstall, A.M. (2012) Reversible jump methods for generalised linear models and generalised linear mixed models. *Statistics and Computing*, **22 (1)**, 107–120.

Gamerman, D. (1997) Sampling from the posterior distribution in generalised linear mixed models. *Statistics and Computing*, **7 (1)**, 57–68.

Nztoufras, I., Dellaportas, P. & Forster, J.J. (2003) Bayesian variable and link determination for generalised linear models. *Journal of Statistical Planning and Inference*, **111 (1)**, 165–180.

Overstall, A.M. & King, R. (2014) conting: An R package for Bayesian analysis of complete and incomplete contingency tables. *Journal of Statistical Software*, **58 (7)**, 1–27. http://www.jstatsoft.org/v58/i07/

Overstall, A.M., King, R., Bird, S.M., Hutchinson, S.J. & Hay, G. (2014) Incomplete contingency tables with censored cells with application to estimating the number of people who inject drugs in Scotland. *Statistics in Medicine*, **33 (9)**, 1564–1579.

See Also

bict.fit, spina, ScotPWID.

Examples

```
set.seed(1)
## Set seed for reproducibility.

data(spina)
## Load the spina data

test1<-bict(formula=y~(S1 + S2 + S3 + eth)^2,data=spina,n.sample=50, prior="UIP")
## Let the maximal model be the model with two-way interactions. Starting from the
## posterior mode of the model with two-way interactions, do 50 iterations under the
## unit information prior.

test1<-bictu(object=test1,n.sample=50)
## Do another 50 iterations

test1

#Number of cells in table = 24
#
#Maximal model =
#y ~ (S1 + S2 + S3 + eth)^2
#
#Number of log-linear parameters in maximal model = 15
#
#Number of MCMC iterations = 100
#
#Computer time for MCMC = 00:00:01
#
```

```
#Prior distribution for log-linear parameters = UIP
#
#Number of missing cells = 3
#
#Number of censored cells = 0

summary(test1)
## Summarise the result. Will get:

#Posterior summary statistics of log-linear parameters:
#            post_prob post_mean post_var lower_lim upper_lim
#(Intercept)         1    1.0427 0.033967    0.6498    1.4213
#S11                 1   -0.3159 0.015785   -0.4477   -0.1203
#S21                 1    0.8030 0.018797    0.6127    1.1865
#S31                 1    0.7951 0.003890    0.6703    0.8818
#eth1                1    2.8502 0.033455    2.4075    3.1764
#eth2                1    0.1435 0.072437   -0.4084    0.5048
#S21:S31             1   -0.4725 0.002416   -0.5555   -0.3928
#NB: lower_lim and upper_lim refer to the lower and upper values of the
#95 % highest posterior density intervals, respectively
#
#Posterior model probabilities:
#  prob model_formula
#1 0.36 ~S1 + S2 + S3 + eth + S2:S3
#2 0.19 ~S1 + S2 + S3 + eth + S2:S3 + S2:eth
#3 0.12 ~S1 + S2 + S3 + eth + S1:eth + S2:S3
#4 0.12 ~S1 + S2 + S3 + eth + S1:S2 + S1:S3 + S1:eth + S2:S3 + S2:eth + S3:eth
#5 0.10 ~S1 + S2 + S3 + eth + S1:S3 + S1:eth + S2:S3
#6 0.06 ~S1 + S2 + S3 + eth + S1:S3 + S1:eth + S2:S3 + S2:eth
#
#Total number of models visited =  8
#
#Posterior mean of total population size = 726.75
#95 % highest posterior density interval for total population size = ( 706 758 )
#
#Under the X2 statistic
#
#Summary statistics for T_pred
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  8.329  15.190  20.040  22.550  24.180 105.200
#
#Summary statistics for T_obs
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  5.329  18.270  22.580  21.290  24.110  37.940
#
#Bayesian p-value =  0.45
```

Description

This function is the workhorse behind bict and bictu.

Usage

```
bict.fit(priornum, missing1, missing2, maximal.mod, IP, eta.hat, ini.index,
ini.beta, ini.sig, ini.y0, iters, save, name, null.move.prob, a, b, progress)
```

Arguments

priornum	A numeric scalar indicating which prior is to be used: 1 = "UIP", 2 = "SBH".
missing1	A vector of the same length as the number of missing cell counts giving the row numbers of the data.frame in data which correspond to the missing cell counts.
missing2	A vector of the same length as the number of censored cell counts giving the row numbers of the data.frame in data which correspond to the censored cell counts.
maximal.mod	An object of class "glm" giving the fit of the maximal model.
IP	A p by p matrix giving the inverse of the prior scale matrix for the maximal model.
eta.hat	A vector of length n (number of cells) giving the posterior mode of the linear predictor under the maximal model.
ini.index	A binary vector, of the same length as the number of log-linear parameters in the maximal model, indicating which parameters are present in the initial model.
ini.beta	A numeric vector giving the starting values of the log-linear parameters for the MCMC algorithm.
ini.sig	A numeric scalar giving the starting value of sigma^2 for the MCMC algorithm.
ini.y0	A numeric vector giving the starting values of the missing and censored cell entries for the MCMC algorithm.
iters	The number of iterations of the MCMC algorithm to peform.
save	If positive, the function will save the MCMC output to external text files every save iterations. If zero , the function will not save the MCMC output to external files.
name	A prefix to the external files saved if the argument save is positive. If NULL, then the external files will have no prefix.
null.move.prob	A scalar argument giving the probability of performing a null move, i.e. proposing a move to the current model.
a	The shape hyperparameter of the Sabanes-Bove & Held prior, see Overstall & King (2014).
b	The scale hyperparameter of the Sabanes-Bove & Held prior, see Overstall & King (2014).
progress	Logical argument. If TRUE, then a progress bar will be displayed.

Value

The function will return a list with the following components.

BETA	An iters by p matrix containing the sampled values of the log-linear parameters, where p is the number of log-linear parameters in the maximal model. For elements of this matrix which correspond to a log-linear parameter which is not present for the current model a zero is returned.
MODEL	A vector of length iters giving the sampled model indicators in hexadecimal format.
SIG	A vector of length iters giving the sampled values for sigma^2 under the Sabanes-Bove & Held prior. If the unit information prior is used then the components of this vector will be one.
Y0	An iters by k matrix giving the sampled values of the missing and censored cell counts, where k is the total number of missing and censored cell counts.
rj_acc	A binary vector of the same length as the number of reversible jump moves attempted. A 0 indicates that the proposal was rejected, and a 1 that the proposal was accepted.
mh_acc	A binary vector of the same length as the number of Metropolis-Hastings moves attempted. A 0 indicates that the proposal was rejected, and a 1 that the proposal was accepted.

Note

This function will not typically be called by the user.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

References

Overstall, A.M. & King, R. (2014) conting: An R package for Bayesian analysis of complete and incomplete contingency tables. *Journal of Statistical Software*, **58 (7)**, 1–27. http://www.jstatsoft.org/v58/i07/

See Also

bict, bictu.

Examples

```
data(spina)
## Load spina data.

spina$z<-spina$y
spina$z[is.na(spina$y)]<-0
## Define a new variable in spina data.frame which is equal to y except where
## y is NA, in which case z=0. This is just so we can fit maximal model to the
```

```
## complete contingency table.

maximal.mod<-glm(formula=z~(S1+S2+S3+eth)^2,data=spina,x=TRUE,y=TRUE,
contrasts=list(S1="contr.sum",S2="contr.sum",S3="contr.sum",
eth="contr.sum"))
## Fit maximal model to complete contingency table.

curr.index<-formula2index(big.X=maximal.mod$x,formula=z~S1+S2+S3+eth,data=spina)
## Set up binary vector for independence model.

IP<-t(maximal.mod$x)%*%maximal.mod$x/length(maximal.mod$y)
IP[,1]<-0
IP[1,]<-0
## Set up the inverse scale matrix for the prior distribution under
## the maximal model.

bmod<-beta_mode(X=maximal.mod$x[!is.na(spina$y),],prior="UIP",
y=maximal.mod$y[!is.na(spina$y)],IP=IP)
## Find the posterior mode under the maximal model fitted to observed cell counts.
eta.hat<-as.vector(maximal.mod$x%*%bmod)
## Find the posterior mode of the linear predictor
## under the maximal model.

set.seed(1)
## Set seed for reproducibility
test1<-bict.fit(priornum=1, missing1=(1:length(maximal.mod$y))[is.na(spina$y)],
missing2=NULL,maximal.mod=maximal.mod, IP=IP, eta.hat=eta.hat, ini.index=curr.index,
ini.beta=bmod[curr.index==1], ini.sig=1, ini.y0=c(500,200,20),iters=10, save=0, name=NULL,
null.move.prob=0.5, a=0.001, b=0.001, progress = FALSE)
## Run for 10 iterations starting at model defined by curr.index.
test1$MODEL
## Look at sampled model indicators. Should be:
#  [1] "7e00" "7e00" "7e00" "7e00" "7e00" "7e00" "7e00" "7e00" "7f00" "7f00"

model2index(test1$MODEL,dig=15)
## Convert these to binary indicators of the log-linear parameters.
## Will get:

#     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
#7e00    1    1    1    1    1    1    0    0    0     0     0
#7e00    1    1    1    1    1    1    0    0    0     0     0
#7e00    1    1    1    1    1    1    0    0    0     0     0
#7e00    1    1    1    1    1    1    0    0    0     0     0
#7e00    1    1    1    1    1    1    0    0    0     0     0
#7e00    1    1    1    1    1    1    0    0    0     0     0
#7e00    1    1    1    1    1    1    0    0    0     0     0
#7e00    1    1    1    1    1    1    0    0    0     0     0
#7f00    1    1    1    1    1    1    1    0    0     0     0
#7f00    1    1    1    1    1    1    1    0    0     0     0
#     [,12] [,13] [,14] [,15]
#7e00     0     0     0     0
#7e00     0     0     0     0
#7e00     0     0     0     0
```

```
#7e00    0    0    0    0
#7e00    0    0    0    0
#7e00    0    0    0    0
#7e00    0    0    0    0
#7e00    0    0    0    0
#7f00    0    0    0    0
#7f00    0    0    0    0
```

`find_cens` *Find Censored Cells*

Description

Given all the sources and the censored source of an incomplete contingency table, this function will find the censored cells.

Usage

```
find_cens(sources, cens_source, data=NULL, unobs.level = "un", obs.level = "obs")
```

Arguments

`sources`	An object of class `"formula"`, which details the sources in the incomplete contingency table.
`cens_source`	An object of class `"formula"`, which details the source which is subject to censoring in the incomplete contingency table.
`data`	An object of class `"data.frame"` (or `"table"`) containing the variables in the model. If the model variables are not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `find_cens` is called.
`unobs.level`	The character string used to label the source level corresponding to not observing the individuals in the cell.
`obs.level`	The character string used to label the source level corresponding to observing the individuals in the cell.

Details

Sometimes one of the sources (termed the censored source) used to estimate closed populations observes individuals which are not members of the target population. In this case we assume that when this source observes an individual that has been observed by at least one other source, then it is a member of the target population. However those individuals only observed by the censored source contain a mixture of members of the target and non-target populations. This means that the observed cell count acts as an upper bound on the true cell count. For more details on this approach, see Overstall et al (2014) and Overstall & King (2014). This function identifies the cells which are censored (i.e. correspond to only being observed by the censored source).

Value

The function will output a numeric vector containing the cell numbers of the censored cells. These are used by the bict and bictu functions.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

References

Overstall, A.M., King, R., Bird, S.M., Hutchinson, S.J. & Hay, G. (2014) Incomplete contingency tables with censored cells with application to estimating the number of people who inject drugs in Scotland. *Statistics in Medicine*, **33 (9)**, 1564–1579.

Overstall, A.M. & King, R. (2014) conting: An R package for Bayesian analysis of complete and incomplete contingency tables. *Journal of Statistical Software*, **58 (7)**, 1–27. http://www.jstatsoft.org/v58/i07/

See Also

bict.

Examples

```
data(ScotPWID)
## Load the ScotPWID data. In this dataset, the S4 source corresponding
## to the HCV database is subject to censoring. We use find_cens to find
## the censored cells.
find_cens(sources=~S1+S2+S3+S4,cens_source=~S4,data=ScotPWID)
## It will produce the vector with the following elements:
##[1]   9  25  41  57  73  89 105 121
## Let's look at these cells
ScotPWID[find_cens(sources=~S1+S2+S3+S4,cens_source=~S4,data=ScotPWID),]
## It will produce:
#      y S1 S2 S3  S4 Region Gender   Age
#9   122 un un un obs    GGC   Male Young
#25  135 un un un obs    GGC   Male   Old
#41   48 un un un obs    GGC Female Young
#57   38 un un un obs    GGC Female   Old
#73  134 un un un obs   Rest   Male Young
#89  104 un un un obs   Rest   Male   Old
#105  78 un un un obs   Rest Female Young
#121  25 un un un obs   Rest Female   Old
```

formula2index *Convert Between Formula and Index*

Description

These functions will convert a formula object to a binary index and vice versa.

Usage

```
formula2index(big.X, formula, data)

index2formula(index, maximal.mod)
```

Arguments

big.X	The design matrix under the maximal model.
formula	An object of class "formula": a symbolic description of the model to convert to a binary index.
data	An object of class "data.frame" containing the variables in the model.
maximal.mod	An object of class "glm" giving the fit of the maximal model.
index	A binary vector, of the same length as the number of log-linear parameters in the maximal model, indicating which parameters are present in the model to be converted to a formula.

Value

The function formula2index will produce a binary vector of the same length as the number of log-linear parameters in the maximal model, indicating which parameters are present in the model represented by the argument formula.

The function index2formula will produce an object of class "formula": a symbolic description of the model given by the argument index.

Note

This function will not typically be called by the user.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

Examples

```
data(ScotPWID)
## Load the ScotPWID data

maximal.mod<-glm(y~(S1+S2+S3+S4+Region+Gender+Age)^2,family=poisson,contrasts=list(
S1="contr.sum",S2="contr.sum",S3="contr.sum",S4="contr.sum",
Region="contr.sum",Gender="contr.sum",Age="contr.sum"),data=ScotPWID,x=TRUE)
## Fit the maximal model containing all two-way interactions.

big.X<-maximal.mod$x
## Set the design matrix under the maximal model

index<-formula2index(big.X=big.X,
formula=~S1+S2+S3+S4+Region+Gender+Age+S1:S2+S1:Age+S2:Gender+S3:S4+S4:Age,
data=ScotPWID)
## Find the index under the model with the following interactions:
## S1:S2
## S1:Age
## S2:Gender
## S3:S4
## S4:Age

index
## Print index
# [1] 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0

index2formula(index=index,maximal.mod=maximal.mod)
## Go back to formula
#y ~ S1 + S2 + S3 + S4 + Region + Gender + Age + S1:S2 + S1:Age +
#    S2:Gender + S3:S4 + S4:Age
```

`heart` *Risk Factors for Coronary Heart Disease: A Complete 2^6 Table*

Description

1841 men are cross-classified according to six risk factors for coronary heart disease: smoking (`A`; 2 levels), strenuous mental work (`B`; 2 levels), strenuous physical work (`C`; 2 levels), systolic blood pressure (`D`; 2 levels), ratio of alpha and beta lipoproteins (`E`; 2 levels) and family anamnesis of coronary heart disease (`F`; 2 levels).

Usage

```
data(heart)
```

Format

A `"data.frame"` with 64 observations on the following 7 variables.

`y` Counts in each cell of table.

A A factor with levels yes no indicating smoking status.

B A factor with levels yes no indicating strenuous mental work.

C A factor with levels yes no indicating strenuous physical work.

D A factor with levels yes no indicating systolic blood pressure.

E A factor with levels yes no indicating high ratio of alpha and beta lipoproteins.

F A factor with levels yes no indicating a family anamnesis of coronary heart disease.

Details

For more details on this data see Edwards & Havranek (1985).

For details on the function bcct applied to this data, see Overstall & King (2014).

Source

Edwards, D. & Havranek, T. (1985) A fast procedure for model search in multidimensional contingency tables. *Biometrika*, **72 (2)**, 339–351.

References

Overstall, A.M. & King, R. (2014) conting: An R package for Bayesian analysis of complete and incomplete contingency tables. *Journal of Statistical Software*, **58 (7)**, 1–27. http://www.jstatsoft.org/v58/i07/

Examples

```
data(heart)
summary(heart)
```

---

index2model *Convert Between Index and Model Indicator*

Description

These functions convert the binary vector, indicating which terms are in the current model, to the hexadecimal model indicator, and vice versa.

Usage

```
index2model(index)

model2index(model,dig)
```

Arguments

index	A binary vector, of the same length as the number of log-linear parameters in the maximal model, indicating which parameters are present in the model to be converted to a hexadecimal.
dig	A scalar argument giving the number of columns of the design matrix for the maximal model.
model	A character string giving a hexadecimal model indicator.

Value

index2model will return a hexadecimal model indicator.

model2index will return a binary vector, of the same length as the number of log-linear parameters in the maximal model, indicating which parameters are present in the model converted from hexadecimal.

Note

This function will not typically be called by the user.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

Examples

```
data(ScotPWID)
## Load the ScotPWID data

maximal.mod<-glm(y~(S1+S2+S3+S4+Region+Gender+Age)^2,family=poisson,contrasts=list(
S1="contr.sum",S2="contr.sum",S3="contr.sum",S4="contr.sum",
Region="contr.sum",Gender="contr.sum",Age="contr.sum"),data=ScotPWID,x=TRUE)
## Fit the maximal model containing all two-way interactions.

big.X<-maximal.mod$x
## Set the design matrix under the maximal model

index<-formula2index(big.X=big.X,
formula=~S1+S2+S3+S4+Region+Gender+Age+S1:S2+S1:Age+S2:Gender+S3:S4+S4:Age,
data=ScotPWID)
## Find the index under the model with the following interactions:
## S1:S2
## S1:Age
## S2:Gender
## S3:S4
## S4:Age

index
## Print the index, will get:
# [1] 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0
```

```
modind<-index2model(index)
## Find the hexadecimal model indicator
modind
## Print it, will get:
#[1] "1ff08a08"

## Convert back to index
model2index(model=modind,dig=length(index))
## Will get:
# [1] 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0
```

`inter_probs` *Calculate Posterior Probability of Each Term*

Description

This function computes the posterior probability of each term using the MCMC output of `"bcct"` and `"bict"` objects.

Usage

```
inter_probs(object, cutoff = 0.75, n.burnin = 0, thin = 1)
```

Arguments

`object`	An object of class `"bcct"` or `"bict"`.
`cutoff`	An optional argument giving the cutoff posterior probability for displaying posterior summary statistics of the log-linear parameters. Only those log-linear parameters with a posterior probability greater than `cutoff` will be returned as part of the output. The default value is 0.75.
`n.burnin`	An optional argument giving the number of iterations to use as burn-in. The default value is 0.
`thin`	An optional argument giving the amount of thinning to use, i.e. the computations are based on every `thin`-th value in the MCMC sample. The default value is 1, i.e. no thinning.

Details

This function provides a scaled back version of what `inter_stats` provides.

The use of thinning is recommended when the number of MCMC iterations and/or the number of log-linear parameters in the maximal model are/is large, which may cause problems with comuter memory storage.

Value

This function returns an object of class "interprob" which is a list with the following components.

term	A vector of term labels.
prob	A vector of posterior probabilities.
thin	The value of the argument thin.

The function will only return elements in the above list if prob > cutoff.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

See Also

bcct, bict, print.interprob, inter_stats.

Examples

```
set.seed(1)
## Set seed for reproducibility
data(AOH)
## Load AOH data

test1<-bcct(formula=y~(alc+hyp+obe)^3,data=AOH,n.sample=100,prior="UIP")
## Starting from maximal model of saturated model do 100 iterations of MCMC
## algorithm.

inter_probs(test1,n.burnin=10,cutoff=0)
## Calculate posterior probabilities having used a burn-in phase of
## 10 iterations and a cutoff of 0 (i.e. display all terms with
## non-zero posterior probability). Will get the following:

#Posterior probabilities of log-linear parameters:
#            post_prob
#(Intercept)    1.0000
#alc            1.0000
#hyp            1.0000
#obe            1.0000
#alc:hyp        0.1778
#alc:obe        0.0000
#hyp:obe        0.4444
#alc:hyp:obe    0.0000

## Note that the MCMC chain (after burn-in) does not visit any models
## with the alc:obe or alc:hyp:obe interactions.
```

inter_stats *Compute Posterior Summary Statistics of the Log-Linear Parameters.*

Description

This function computes the posterior summary statistics of the log-linear parameters using the MCMC output of "bcct" and "bict" objects. The posterior summary statistics are posterior probability, posterior mean, posterior variance and lower and upper limits highest posterior density intervals (HPDIs).

Usage

```
inter_stats(object, cutoff = 0.75, n.burnin = 0, thin = 1, prob.level = 0.95)
```

Arguments

object	An object of class "bcct" or "bict".
cutoff	An optional argument giving the cutoff posterior probability for displaying posterior summary statistics of the log-linear parameters. Only those log-linear parameters with a posterior probability greater than cutoff will be returned as part of the output. The default value is 0.75.
n.burnin	An optional argument giving the number of iterations to use as burn-in. The default value is 0.
thin	An optional argument giving the amount of thinning to use, i.e. the computations are based on every thin-th value in the MCMC sample. The default value is 1, i.e. no thinning.
prob.level	An optional argument giving the probability content of the HPDIs. The default value is 0.95.

Details

This function provides an expanded version of what inter_probs provides.

The use of thinning is recommended when the number of MCMC iterations and/or the number of log-linear parameters in the maximal model are/is large, which may cause problems with comuter memory storage.

Value

This function will return an object of class "interstat" which is a list with the following components:

term	A vector of term labels for each parameter.
prob	A vector of posterior probabilities for each parameter.
post_mean	A vector of posterior means for each parameter.
post_var	A vector of posterior variances for each parameter.

lower	A vector of lower limits for the 100*prob.level% HPDI for each parameter.
upper	A vector of upper limits for the 100*prob.level% HPDI for each parameter.
prob.level	The argument prob.level.

The function will only return elements in the above list if prob > cutoff.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

See Also

bcct, bict, print.interstat inter_probs

Examples

```
set.seed(1)
## Set seed for reproducibility
data(AOH)
## Load AOH data

test1<-bcct(formula=y~(alc+hyp+obe)^3,data=AOH,n.sample=100,prior="UIP")
## Starting from maximal model of saturated model do 100 iterations of MCMC
## algorithm.

inter_stats(test1,n.burnin=10,cutoff=0.5)
## Calculate posterior summary statistics having used a burn-in phase of
## 10 iterations and a cutoff of 0 (i.e. display all terms with
## non-zero posterior probability. Will get the following:

#Posterior summary statistics of log-linear parameters:
#            post_prob post_mean post_var lower_lim upper_lim
#(Intercept)         1   2.88291 0.002565   2.78778   2.97185
#alc1                1  -0.05246 0.008762  -0.27772   0.06655
#alc2                1  -0.05644 0.006407  -0.20596   0.11786
#alc3                1   0.06822 0.005950  -0.09635   0.18596
#hyp1                1  -0.53895 0.003452  -0.63301  -0.39888
#obe1                1  -0.04686 0.007661  -0.20929   0.12031
#obe2                1   0.01395 0.004024  -0.11024   0.11783
#NB: lower_lim and upper_lim refer to the lower and upper values of the
#95 % highest posterior density intervals, respectively
```

---

iwls_mh *Iterated Weighted Least Square Metropolis Hastings Algorithm*

Description

This function implements one iteration of the Iterated Weight Least Square Metropolis Hastings Algorithm as proposed by Gamerman (1997) for generalised linear models as applied to log-linear models.

Usage

```
iwls_mh(curr.y, curr.X, curr.beta, iprior.var)
```

Arguments

curr.y	A vector of length n giving the cell counts.
curr.X	An n by p design matrix for the current model, where p is the number of log-linear parameters.
curr.beta	A vector of length p giving the current log-linear parameters.
iprior.var	A p by p matrix giving the inverse of the prior variance matrix.

Details

For details of the original algorithm see Gamerman (1997). For its application to log-linear models see Overstall & King (2014), and the references therein.

Value

The function will output a vector of length p giving the new values of the log-linear parameters.

Note

This function will not typically be called by the user.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

References

Gamerman, D. (1997) Sampling from the posterior distribution in generalised linear mixed models. *Statistics and Computing*, **7 (1)**, 57–68.

Overstall, A.M. & King, R. (2014) conting: An R package for Bayesian analysis of complete and incomplete contingency tables. *Journal of Statistical Software*, **58 (7)**, 1–27. http://www.jstatsoft.org/v58/i07/

Examples

```
set.seed(1)
## Set seed for reproducibility
data(AOH)
## Load AOH data

maximal.mod<-glm(y~alc+hyp+obe,family=poisson,x=TRUE,contrasts=list(alc="contr.sum",
hyp="contr.sum",obe="contr.sum"),data=AOH)
## Fit independence model to get a design matrix

IP<-t(maximal.mod$x)%*%maximal.mod$x/length(AOH$y)
IP[,1]<-0
```

```
IP[1,]<-0
## Set up inverse prior variance matrix under the UIP

## Let the current parameters be the MLE under the independence model
as.vector(coef(maximal.mod))
#[1]  2.89365105 -0.04594959 -0.07192507  0.08971628 -0.50545335  0.00818037
#[7] -0.01636074

## Update parameters using MH algorithm
iwls_mh(curr.y=AOH$y,curr.X=maximal.mod$x,curr.beta=coef(maximal.mod),iprior.var=IP)

## Will get:
#[1]  2.86468919 -0.04218623 -0.16376055  0.21656167 -0.49528676 -0.05026597
#[7]  0.02726671
```

---

mod_probs *Compute Posterior Model Probabilities*

Description

This function computes the posterior model probabilities using the MCMC output of "bcct" and "bict" objects.

Usage

```
mod_probs(object, n.burnin = 0, scale = 0.1, best = NULL, thin = 1)
```

Arguments

object	An object of class "bcct" or "bict".
n.burnin	An optional argument giving the number of iterations to use as burn-in. The default value is 0.
scale	An optional argument for controlling how the posterior model probabilities are returned as output. The function will return details on the models with the posterior model probability larger than scale times the probability of the posterior modal model. The default value is 0.1.
best	An optional argument for controlling how the posterior model probabilities are returned as output. The function will return details on the best models with the highest posterior model probabilities. For example, if best=4, then details on the four models with the highest posterior model probabilities will be returned. The default value is NULL. If not NULL, then this argument takes precedent over scale.
thin	An optional argument giving the amount of thinning to use, i.e. the computations are based on every thin-th value in the MCMC sample. The default value is 1, i.e. no thinning.

Details

It will output only the probabilities of the "best" models, as defined by the user specifying either the `best` or `scale` arguments.

The use of thinning is recommended when the number of MCMC iterations and/or the number of log-linear parameters in the maximal model are/is large, which may cause problems with comuter memory storage.

Value

The function will return an object of class `"modprobs"` which is a list containing the following components.

`table`	An object of class `"data.frame"` with number of rows defined by `scale` or `best` and columns: `model_formula`; giving the model (in terms of a printed formula), and `prob`; giving the posterior model probability.
`totmodsvisit`	A numeric scalar giving the total number of models visited after the burn-in iterations.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

See Also

`bcct`, `bict`, `print.modprobs`.

Examples

```
set.seed(1)
## Set seed for reproducibility
data(AOH)
## Load AOH data

test1<-bcct(formula=y~(alc+hyp+obe)^3,data=AOH,n.sample=100,prior="UIP")
## Starting from maximal model of saturated model do 100 iterations of MCMC
## algorithm.

mod_probs(object=test1,n.burnin=10,best=6)
## Using a burn-in of 10 iterations find the posterior model probabilities
## of the 6 models with the highest posterior model probability. Will get:

#Posterior model probabilities:
#  prob    model_formula
#1 0.50000 ~alc + hyp + obe
#2 0.32222 ~alc + hyp + obe + hyp:obe
#3 0.12222 ~alc + hyp + obe + alc:hyp + hyp:obe
#4 0.05556 ~alc + hyp + obe + alc:hyp
#
#Total number of models visited =  4
```

```
## Note that since the chain only visited 4 models we only get probabilities
## for 4 models not 6.
```

`plot.pval` *Plot* `pval` *Objects*

Description

This function plots objects of class `"pval"`.

Usage

```
## S3 method for class 'pval'
plot(x, ...)
```

Arguments

`x`	An object of class `"pval"`.
`...`	Arguments to be passed to and from other methods.

Value

This function will produce a plot of `T_obs` against `T_pred` (see `bayespval`) along with a line through the origin with slope one. The proportion of points above the line (grey points) gives the Bayesian p-value.

Note

For an example see `bayespval`.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

See Also

`bayespval`

plot.totpop *Plot* totpop *Objects*

Description

This function plots objects of class "totpop".

Usage

```
## S3 method for class 'totpop'
plot(x, ...)
```

Arguments

x	An object of class "totpop".
...	Arguments to be passed to and from other methods.

Value

This function will produce a histogram of the MCMC sample from the posterior distribution of the total population size.

Note

For an example see total_pop.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

See Also

total_pop

print.acceptrate *Prints* acceptrate *Objects*

Description

This function prints objects of class "acceptrate".

Usage

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

Arguments

x	An object of class "acceptrate".
digits	An optional argument controlling the rounding of output.
...	Arguments to be passed to and from other methods.

Value

This function will simply print out the acceptance rates for the reversible jump and Metropolis-Hastings algorithms.

Note

For an example see accept_rate.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

See Also

accept_rate

---

print.bcct *Print* bcct *and* bict *Objects*

Description

This function prints objects of class "bcct" and "bict".

Usage

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

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

Arguments

x	An object of class "bcct" or "bict".
...	Arguments to be passed to and from other methods.

Value

These functions print out very simple details on the bcct or bict objects. They display the number of cells in the table, the maximal model considered, the number of log-linear parameters in the maximal model, the number of MCMC iterations, the computer time required for the MCMC (in hours, minutes and seconds) and the prior used.

In the case of objects of class "bict", it also prints out the number of missing and censored cells.

Note

For examples see bcct and bict.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

See Also

bcct, bict.

---

print.interprob *Print* interprob *Objects*

Description

This function prints objects of class "interprob".

Usage

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

Arguments

x	An object of class "interprob".
digits	An optional argument controlling the rounding of output.
...	Arguments to be passed to and from other methods.

Value

This function will print out the posterior probability of each term (subject to the argument cutoff).

Note

For an example see inter_probs.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

See Also

inter_probs.

---

print.interstat *Print* interstat *Objects*

Description

This function prints objects of class "interstat".

Usage

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

Arguments

x	An object of class "interstat".
digits	An optional argument controlling the rounding of output.
...	Arguments to be passed to and from other methods.

Value

This function will print out the posterior probability, posterior mean, posterior variance and the 100*prob.level% highest posterior density intervals (HPDIs) of each log-linear parameter (subject to the argument cutoff).

Note

For an example see inter_stats.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

See Also

inter_stats.

print.modprobs *Print* modprobs *Objects*

Description

This function prints objects of class "modprobs".

Usage

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

Arguments

x	An object of class "modprobs".
digits	An optional argument controlling the rounding of output.
...	Arguments to be passed to and from other methods.

Value

This function will print out the posterior model probability of the "best" models as defined by the arguments best or scale.

Note

For an example see mod_probs.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

See Also

mod_probs.

print.pval *Print* pval *Objects*

Description

This function prints objects of class "pval".

Usage

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

Arguments

x	An object of class "pval".
digits	An optional argument controlling the rounding of output.
...	Arguments to be passed to and from other methods.

Value

This function will print out summary statistics for the predictive (T_pred) and observed (T_obs) discrepancy statistics. Additionally it will output the associated Bayesian p-value.

Note

For an example see bayespval.

Author(s)

Antony M. Overstall (<antony@mcs.st-and.ac.uk>).

See Also

bayespval

---

print.submod *Print* submod *Objects*

Description

This function prints objects of class "submod".

Usage

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

Arguments

x	An object of class "submod".
...	Arguments to be passed to and from other methods.
digits	An optional argument controlling the rounding of output.

Value

Firstly, conditional on the model of interest (defined by formula and order), this function will print out the posterior means, posterior variances and 100*prob.level% highest posterior density intervals (HPDIs) for each of the log-linear parameters. Secondly, conditional on the model of interest, it will print out summaries of the discrepancy statistics and the corresponding Bayesian p-value. Finally, if the class of the object passed to sub_model is "bict", then it will print out the posterior mean and 100*prob.level% HPDI for the total population size, conditional on the model of interest.

Note

For an example see sub_model.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

See Also

sub_model.

---

print.totpop *Print* totpop *Objects*

Description

This function prints objects of class "totpop".

Usage

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

Arguments

x	An object of class "totpop".
digits	An optional argument controlling the rounding of output.
...	Arguments to be passed to and from other methods.

Value

The function will print out the posterior mean and the 100*prob.level% highest posterior density interval for the total population size.

Note

For an example see total_pop.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

See Also

total_pop.

RJ_update *Reversible Jump Algorithm*

Description

This function implements one iteration of the orthogonal projection reversible jump algorithm for generalised linear models proposed by Forster et al (2012) applied to log-linear models.

Usage

```
RJ_update(prop.index, curr.index, curr.beta, eta.hat, curr.y, big.X,
proposal.probs, i.prop.prior.var, i.curr.prior.var)
```

Arguments

prop.index	A binary vector, of the same length as the number of log-linear parameters in the maximal model, indicating which parameters are present in the proposed model.
curr.index	A binary vector, of the same length as the number of log-linear parameters in the maximal model, indicating which parameters are present in the current model.
curr.beta	A vector of length sum(curr.index) giving the log-linear parameters under the current model.
eta.hat	A vector of length n (number of cells) giving the posterior mode of the linear predictor under the maximal model.
curr.y	A vector of length n giving the cell counts.
big.X	The design matrix under the maximal model.
proposal.probs	A numeric vector of length 2. The first element gives the probability of proposing a move from the proposed model to the current model. The second element gives the probability of proposing a move from the current model to the proposed model.
i.prop.prior.var	A matrix giving the inverse of the prior variance matrix for the log-linear parameters under the proposed model.
i.curr.prior.var	A matrix giving the inverse of the prior variance matrix for the log-linear parameters under the current model.

Details

For the original algorithm see Forster et al (2012). For details on its application to log-linear models see Overstall & King (2014), and the references therein.

Value

The function will return a list with the following components:

new.beta	A vector giving the new log-linear parameters.
new.index	A binary vector indicating which log-linear parameters are present in the new model.

Note

This function will not typically be called by the user.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

References

Forster, J.J., Gill, R.C. & Overstall, A.M. (2012) Reversible jump methods for generalised linear models and generalised linear mixed models. *Statistics and Computing*, **22 (1)**, 107–120.

Overstall, A.M. & King, R. (2014) conting: An R package for Bayesian analysis of complete and incomplete contingency tables. *Journal of Statistical Software*, **58 (7)**, 1–27. http://www.jstatsoft.org/v58/i07/

Examples

```
set.seed(4)
## Set seed for reproducibility
data(AOH)
## Load data
maximal.mod<-glm(y~(alc+hyp+obe)^3,family=poisson,x=TRUE,contrasts=list(alc="contr.sum",
hyp="contr.sum",obe="contr.sum"),data=AOH)
## Fit maximal model to get a design matrix

IP<-t(maximal.mod$x)%*%maximal.mod$x/length(AOH$y)
IP[,1]<-0
IP[1,]<-0
## Calculate inverse prior scale matrix under maximal model. Under the UIP this
## is the inverse prior variance matrix. Under the SBH prior, we need to divide
## this matrix by the current value of SIG.

bmod<-beta_mode(X=maximal.mod$x,y=AOH$y,IP=IP)
## Find posterior mode under maximal model with UIP
eta.hat<-as.vector(maximal.mod$x%*%bmod)
## Find posterior mode of linear predictor.

curr.index<-formula2index(big.X=maximal.mod$x,formula=y~alc+hyp+obe+alc:hyp,data=AOH)
## Calculate index for current model including alc:hyp interaction
curr.index
## Print out current index
#[1] 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
pm<-prop_mod(curr.index=curr.index,data=AOH,maximal.mod=maximal.mod)
## Propose a model
p2<-(1-pm$null.move.prob)/pm$total.choices
p2
## Calculate probability of proposing proposed model from current model
#[1] 0.1666667

prop.index<-pm$new.index
prop.index
## Assign and print out proposal index
# [1] 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

dm<-prop_mod(curr.index=prop.index,data=AOH,maximal.mod=maximal.mod,null.move.prob=0)
p1<-(1-pm$null.move.prob)/dm$total.choices
p1
## Calculate probability of proposing current model from proposed model
#[1] 0.1666667

RJ_update(prop.index=prop.index,curr.index=curr.index,
curr.beta=coef(maximal.mod)[curr.index==1],eta.hat=eta.hat,curr.y=AOH$y,big.X=maximal.mod$x,
proposal.probs=c(p1,p2),
i.prop.prior.var=IP[prop.index==1,prop.index==1],
i.curr.prior.var=IP[curr.index==1,curr.index==1])

## Do one iteration of reversible jump algorithm. Will get:

#$new.beta
#(Intercept)        alc1        alc2        alc3        hyp1        obe1
# 2.87128918 -0.07098006 -0.07221330  0.08748803 -0.51899802 -0.07855115
#       obe2
#-0.02474727
#
#$new.index
# [1] 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

ScotPWID *People Who Inject Drugs in Scotland 2006: An Incomplete 2^7 Table*

Description

5670 people who inject drugs (PWID) in Scotland in 2006 are osberved by four sources: social enquiry reports (S1), hospital records (S2), Scottish drug misuse database (S3) and Hepatitis C virus (HCV) diagnosis database (S4). The PWID are further cross-classified according to three additional factors: region (Region; 2 levels), gender (Gender; 2 levels) and age (Age; 2 levels).

Usage

```
data(ScotPWID)
```

Format

A "data.frame" with 128 observations on the following 8 variables.

y Counts in each cell of the table with NAs for the cells corresponding to not being observed by any of the sources.

S1 A factor with levels un obs indicating whether source S1 observed the PWID.

S2 A factor with levels un obs indicating whether source S2 observed the PWID.

S3 A factor with levels un obs indicating whether source S3 observed the PWID.

S4 A factor with levels un obs indicating whether source S4 observed the PWID.

Region A factor with levels GGC Rest indicating the region (GGC = Greater Glasgow & Clyde, Rest = Rest of Scotland).

Gender A factor with levels Male Female indicating gender.

Age A factor with levels Young Old indicating age (Young = <35 years, Old=35+ years).

Details

Note that the PWID observed by source S4, the HCV database, are not necessarily current PWID. They are people who have a history of drug use. Therefore the count in the cell corresponding to only being observed by the HCV database is an overcount. Overstall et al (2014) use a modelling approach whereby the count in the cell corresponding to only being observed by the HCV database is missing and the observed value acts as an upper bound. For more details on the dataset see King et al (2013).

For details on the function bict applied to this data, see Overstall & King (2014).

Source

King, R., Bird, S. M., Overstall, A. M., Hay, G. & Hutchinson, S. J. (2013) Injecting drug users in Scotland, 2006: Listing, number, demography, and opiate-related death-rates. *Addiction Research and Theory*, **21 (3)**, 235-246.

References

Overstall, A.M., King, R., Bird, S.M., Hutchinson, S.J. & Hay, G. (2014) Incomplete contingency tables with censored cells with application to estimating the number of people who inject drugs in Scotland. *Statistics in Medicine*, **33 (9)**, 1564–1579.

Overstall, A.M. & King, R. (2014) conting: An R package for Bayesian analysis of complete and incomplete contingency tables. *Journal of Statistical Software*, **58 (7)**, 1–27. http://www.jstatsoft.org/v58/i07/

Examples

```
data(ScotPWID)
summary(ScotPWID)
```

spina *Persons born with Spina Bifida: An Incomplete 2 * 2 * 2 * 3 Table*

Description

621 people born with Spina Bifida (a congenital disorder) in the state of New York between 1969 and 1974 are observed by three sources: birth certificates (S1); death certificates (S2); and medical rehabilitation lists (S3). The people are also cross-classified according to their ethnicity (eth; 3 levels).

Usage

```
data(spina)
```

Format

A "data.frame" with 24 observations on the following 5 variables.

- y Counts in each cell of the table with NAs for the cells corresponding to not being observed by any of the sources.
- S1 A factor with levels un obs indicating whether the birth certificate source observed the person.
- S2 A factor with levels un obs indicating whether the death certificate source observed the person.
- S3 A factor with levels un obs indicating whether the medical rehabilitation source observed the person.
- eth A factor with levels afro-american caucasian codeother indicating the ethnicity of the person (afro-american = Afro-American, causcasian = Caucasian, other = Other).

Details

See Madigan & York (1997), and the references therin, for more details on the study.

For details on the function bict applied to this data, see Overstall & King (2014).

Source

Madigan, D. & York, J.C. (1997) Methods for Estimation of the Size of a Closed Population. *Biometrika*, **84 (1)**, 19–31.

References

Overstall, A.M. & King, R. (2014) conting: An R package for Bayesian analysis of complete and incomplete contingency tables. *Journal of Statistical Software*, **58 (7)**, 1–27. http://www.jstatsoft.org/v58/i07/

Examples

```
data(spina)
summary(spina)
```

sub_model *Compute Posterior Summary Statistics for (Sub-) Models*

Description

This function computes posterior summary statistics for (sub-) models using the MCMC output of "bcct" and "bict" objects.

Usage

```
sub_model(object, formula = NULL, order = 1, n.burnin = 0, thin = 1,
prob.level = 0.95, statistic = "X2")
```

Arguments

object	An object of class "bcct" or "bict".
formula	An optional argument of class "formula": a symbolic description of the model of interest. The default value is NULL. If not NULL then this argument takes precedent over order.
order	A scalar argument identifying the model for which to compute summary statistics. The function will compute statistics for the model with the order-th largest posterior model probability. The default value is 1, meaning that, by default, the function will compute summary statistics for the posterior modal model.
n.burnin	An optional argument giving the number of iterations to use as burn-in. The default value is 0.
thin	An optional argument giving the amount of thinning to use, i.e. the computations are based on every thin-th value in the MCMC sample. The default value is 1, i.e. no thinning.
prob.level	An optional argument giving the probability content of the highest posterior density intervals (HPDIs). The default value is 0.95.
statistic	An optional argument giving the discrepancy statistic to use for calculating the Bayesian p-value. It can be one of c("X2","FreemanTukey","deviance") which correspond to the different statistics: "X2" = Chi-squared statistic, "FreemanTukey" = Freeman-Tukey statistic, "deviance" = deviance statistic. See Overstall & King (2014), and references therein, for descriptions of these statistics.

Details

If the MCMC algorithm does not visit the model of interest in the thinned MCMC sample, after burn-in, then an error message will be returned.

The use of thinning is recommended when the number of MCMC iterations and/or the number of log-linear parameters in the maximal model are/is large, which may cause problems with comuter memory storage.

Value

This function will return an object of class "submod" which is a list with the following components. Note that, unless otherwise stated, all components are conditional on the model of interest.

term	A vector of term labels for each log-linear parameter.
post_prob	A scalar giving the posterior model probability for the model of interest.
post_mean	A vector of posterior means for each of the log-linear parameters.
post_var	A vector of posterior variances for each of the log-linear parameters.
lower	A vector of lower limits for the 100*prob.level% HPDI for each log-linear parameter.
upper	A vector of upper limits for the 100*prob.level% HPDI for each log-linear parameter.
prob.level	The argument prob.level.
order	The ranking of the model of interest in terms of posterior model probabilities.
formula	The formula of the model of interest.
BETA	A matrix containing the sampled values of the log-linear parameters, where the number of columns is the number of log-linear parameters in the model of interest.
SIG	A vector containing the sampled values of sigma^2 under the Sabanes-Bove & Held prior. If the unit information prior is used then the components of this vector will be one.

If object is of class "bict", then sub_model will also return the following component.

Y0	A matrix (with k columns) containing the sampled values of the missing and censored cell counts, where k is the total number of missing and censored cell counts.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

References

Overstall, A.M. & King, R. (2014) conting: An R package for Bayesian analysis of complete and incomplete contingency tables. *Journal of Statistical Software*, **58 (7)**, 1–27. http://www.jstatsoft.org/v58/i07/

See Also

bcct, bict,

Examples

```
set.seed(1)
## Set seed for reproducibility.

data(AOH)
## Load the AOH data

test1<-bcct(formula=y~(alc+hyp+obe)^3,data=AOH,n.sample=100,prior="UIP")
## Let the maximal model be the saturated model. Starting from the
## posterior mode of the maximal model do 100 iterations under the unit
## information prior.

test1sm<-sub_model(object=test1,order=1,n.burnin=10)
## Obtain posterior summary statistics for posterior modal model using a
## burnin of 10.

test1sm

#Posterior model probability =  0.5
#
#Posterior summary statistics of log-linear parameters:
#            post_mean post_var lower_lim upper_lim
#(Intercept)  2.907059 0.002311   2.81725   2.97185
#alc1        -0.023605 0.004009  -0.20058   0.06655
#alc2        -0.073832 0.005949  -0.22995   0.10845
#alc3         0.062491 0.006252  -0.09635   0.18596
#hyp1        -0.529329 0.002452  -0.63301  -0.43178
#obe1         0.005441 0.004742  -0.12638   0.12031
#obe2        -0.002783 0.004098  -0.17082   0.07727
#NB: lower_lim and upper_lim refer to the lower and upper values of the
#95 % highest posterior density intervals, respectively
#
#Under the X2 statistic
#
#Summary statistics for T_pred
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  11.07   19.76   23.34   24.47   29.04   50.37
#
#Summary statistics for T_obs
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  30.82   34.78   35.74   36.28   37.45   42.49
#
#Bayesian p-value =  0.0444
```

---

summary.bcct *Summary of* bcct *and* bict *Objects*

Description

These functions produce summaries of objects of class "bcct" and "bict". They also control how these summaries are printed.

Usage

```
## S3 method for class 'bcct'
summary(object, n.burnin = 0, thin = 1, cutoff = 0.75, statistic = "X2",
best = NULL, scale = 0.1, prob.level = 0.95, ...)

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

## S3 method for class 'bict'
summary(object, n.burnin = 0, thin = 1, cutoff = 0.75, statistic = "X2",
best = NULL, scale = 0.1, prob.level = 0.95, ...)

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

Arguments

object	An object of class "bcct" or "bict".
x	An object of class "sbcct" or "sbict" produced as a result of a call to the functions summary.bcct or summary.bict, respectively.
n.burnin	An optional argument giving the number of iterations to use as burn-in. The default value is 0.
thin	An optional argument giving the amount of thinning to use, i.e. the computations are based on every thin-th value in the MCMC sample. The default value is 1, i.e. no thinning.
cutoff	An optional argument giving the cutoff posterior probability for displaying posterior summary statistics of the log-linear parameters. Only those log-linear parameters with a posterior probability greater than cutoff will be returned as part of the output. The default value is 0.75.
statistic	An optional argument giving the discrepancy statistic to use for calculating the Bayesian p-value. It can be one of c("X2","FreemanTukey","deviance") which correspond to the different statistics: "X2" = Chi-squared statistic, "FreemanTukey" = Freeman-Tukey statistic, "deviance" = deviance statistic. See Overstall & King (2014), and references therein, for descriptions of these statistics.
best	An optional argument for controlling how the posterior model probabilities are returned as output. The function will return details on the best models with the highest posterior model probabilities. The default value is NULL. If not NULL than this argument takes precedent over scale.

scale	An optional argument for controlling how the posterior model probabilities are returned as output. The function will return details on the models with the posterior model probability larger than scale times the probability of the posterior modal model. The default value is 0.1.
prob.level	An optional argument giving the probability content of the highest posterior density intervals (HPDIs). The default value is 0.95.
digits	An optional argument controling the rounding of output.
...	Arguments to be passed to and from other methods.

Details

The functions summary.bcct and summary.bict rely on the functions inter_stats, mod_probs, bayespval, and (in the case of summary.bict) total_pop. For extra information about the output from these functions, see the associated help files.

The use of thinning is recommended when the number of MCMC iterations and/or the number of log-linear parameters in the maximal model are/is large, which may cause problems with comuter memory storage.

Value

The function summary.bcct will return an object of class "sbcct" which is a list with the following components.

BETA	An n.sample by p matrix containing the sampled values of the log-linear parameters, where p is the number of log-linear parameters in the maximal model. For elements of this matrix which correspond to a log-linear parameter which is not present for the current model a zero is returned.
MODEL	A vector of length n.sample giving the samlpled model indicators in hexadecimal format.
SIG	A vector of length n.sample giving the sampled values for sigma^2 under the Sabanes-Bove & Held prior. If the unit information prior is used then the components of this vector will be one.
rj_acc	A binary vector of the same length as the number of reversible jump moves attempted. A 0 indicates that the proposal was rejected, and a 1 that the proposal was accepted.
mh_acc	A binary vector of the same length as the number of Metropolis-Hastings moves attempted. A 0 indicates that the proposal was rejected, and a 1 that the proposal was accepted.
priornum	A numeric scalar indicating which prior was used: 1 = "UIP", 2 = "SBH".
maximal.mod	An object of class "glm" giving the fit of the maximal model.
IP	A p by p matrix giving the inverse of the prior scale matrix for the maximal model.
eta.hat	A vector of length n (number of cells) giving the posterior mode of the linear predictor under the maximal model.
save	The argument save.

name	The argument name.
int_stats	A list which contains the same components as an object of class "interstat", i.e. summary statistics for the log-linear parameters, see inter_stats.
mod_stats	A list which contains the same components as an object of class "modprobs", i.e. summary statistics for the posterior model probabilities, see mod_probs.
pval_stats	A list which contains the same components as an object of class "pval", i.e. summary statistics for the posterior model probabilities, see bayespval.

The function summary.bict will return an object of class "sbict" which is a list with the same components as an object of class "sbcct" and the following additional components.

Y0	An n.sample by k matrix giving the sampled values of the missing and censored cell counts, where k is the total number of missing and censored cell counts.
tpop_stats	A list which contains the same components as an object of class "totpop", i.e. posterior summary statistics for the total population, see total_pop.

The functions print.sbcct and print.sbict will print out the MCMC acceptance rates, posterior summary statistics for the log-linear parameters, the posterior model probabilities, the Bayesian p-value and (in the case of print.sbict) posterior summary statistics for the total population size.

Note

For examples see the help files for bcct and bict.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

References

Overstall, A.M. & King, R. (2014) conting: An R package for Bayesian analysis of complete and incomplete contingency tables. *Journal of Statistical Software*, **58 (7)**, 1–27. http://www.jstatsoft.org/v58/i07/

See Also

bcct, bict, accept_rate, bayespval, inter_stats, mod_probs, total_pop.

---

total_pop *Evaluate Posterior Distribution of Total Population Size*

Description

This function uses the MCMC output of a "bict" object to derive an MCMC sample from the posterior distribution of the total population size.

Usage

```
total_pop(object, n.burnin = 0, thin = 1, prob.level = 0.95)
```

Arguments

object	An object of class "bict".
n.burnin	An optional argument giving the number of iterations to use as burn-in. The default value is 0.
thin	An optional argument giving the amount of thinning to use, i.e. the computations are based on every thin-th value in the MCMC sample. The default value is 1, i.e. no thinning.
prob.level	An optional argument giving the target probability content of the highest posterior density intervals for the total population size. The default value is 0.95.

Details

The use of thinning is recommended when the number of MCMC iterations and/or the number of log-linear parameters in the maximal model are/is large, which may cause problems with comuter memory storage.

Value

This function will return an object of class "totpop" which is a list with the following components.

TOT	A vector of length (n.sample-n.burnin) giving the MCMC sample from the posterior distribution of the total population size.
int	The 100*prob.level% highest posterior density interval (HPDI) for the total population size.
meanTOT	The posterior mean of the total population size.
prob.level	The argument prob.level.

Author(s)

Antony M. Overstall <antony@mcs.st-and.ac.uk>.

See Also

bict, print.totpop.

Examples

```
set.seed(1)
## Set seed for reproducibility

data(spina)
## Load spina data

test1<-bict(formula=y~(S1+S2+S3+eth)^2,data=spina,n.sample=100,prior="UIP")
```

```
## For the spina dataset. We do 100 iterations under the unit information
## prior. The maximal model is the model with two-way interactions and we
## start from this model at the posterior model

tp<-total_pop(test1,n.burnin=10)
## Use a burn-in phase of 10 iterations
tp
## Print out results. Will get:

#Posterior mean of total population size = 727.0667
#95 % highest posterior density interval for total population size = ( 706 757 )

## Could do a plot
## Not run: plot(tp)

## Do a summary of MCMC sample from total population size
summary(tp$TOT)
## Will get

#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  697.0   716.2   727.0   727.1   735.8   763.0
```

Index