# Package ‘RxCEcolInf’

March 18, 2019

**Type** Package

**Title** 'R x C Ecological Inference With Optional Incorporation of Survey Information'

**Version** 0.1-4

**Date** 2019-03-17

**Author** D. James Greiner, Paul Baines, and Kevin M. Quinn

**Maintainer** Kevin M. Quinn <kmq@umich.edu>

**Description** Fits the R x C inference model described in Greiner and Quinn (2009). Allows incorporation of survey results.

**Depends** MASS, MCMCpack, mvtnorm, lattice, coda, stats

**License** GPL

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2019-03-18 14:23:30 UTC

## R topics documented:

RxCEcolInf-package . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
Analyze . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
AnalyzeWithExitPoll . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
gendata.ep . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
stlouis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
texas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
Tune . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
TuneWithExitPoll . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25

**Index** **30**

## `RxCEcolInf-package` *RxCEcolInf*

### Description

Fits the R x C ecological inference model described in Greiner and Quinn (2009). Allows the inclusion of survey information.

### Details

| | |
|---|---|
| Package: | RxCEcolInf |
| Type: | Package |
| Version: | 0.1-1 |
| Date: | 2009-04-14 |
| License: | |

The user should place the data from the contingency tables into a dataframe, each line of which represents one table. The function Tune should be called first; this will tune the markov chain monte carlo algorithm used to fit the model. The user feeds the results from Tune into Analyze, which produces the results. Both Tune and Analyze are called using a string that resembles the R formula interface. If a simple random sample is available from certain contingency tables, the user may incorporate this survey using TuneWithExitPoll and AnalyzeWithExitPoll.

### Author(s)

D. James Greiner, Paul D. Baines, \& Kevin M. Quinn

Maintainer: Kevin M. Quinn <kquinn@fas.harvard.edu>

### References

D. James Greiner \& Kevin M. Quinn. 2009. "R x C Ecological Inference: Bounds, Correlations, Flexibility, and Transparency of Assumptions." *J.R. Statist. Soc. A* 172:67-81.

### Examples

```
## Not run:
library(RxCEcolInf)
data(stlouis)
Tune.stlouis <- Tune("Bosley, Roberts, Ribaudo, Villa, NoVote ~ bvap, ovap",
                     data = stlouis,
                     num.iters = 10000,
                     num.runs = 15)
Chain1.stlouis <- Analyze("Bosley, Roberts , Ribaudo,
                          Villa, NoVote ~ bvap, ovap",
                          rho.vec = Tune.stlouis$rhos,
                          data = stlouis,
                          num.iters = 1500000,
                          burnin = 150000,
                          save.every = 1500,
                          print_every = 15000,
                          debug = 1,
                          keepNNinternals = 100,
                          keepTHETAS = 100)
Chain2.stlouis <- Analyze(”Bosley, Roberts , Ribaudo, Villa,
                          NoVote ~ bvap, ovap”,
                          rho.vec = Tune.stlouis$rhos,
                          data = stlouis,
                          num.iters = 1500000,
                          burnin = 150000,
                          save.every = 1500,
                          print_every = 15000,
                          debug = 1,
                          keepNNinternals = 100,
                          keepTHETAS = 100)
Chain3.stlouis <- Analyze(”Bosley, Roberts , Ribaudo, Villa,
                  NoVote ~ bvap, ovap”,
                          rho.vec = Tune.stlouis$rhos,
                          data = stlouis,
                          num.iters = 1500000,
                          burnin = 150000,
                          save.every = 1500,
                          print_every = 15000,
                          debug = 1,
                          keepNNinternals = 100,
                          keepTHETAS = 100)
stlouis.MCMClist <- mcmc.list(Chain1.stlouis, Chain2.stlouis,
Chain3.stlouis)

## End(Not run)
```

## Analyze *Workhorse Function for Ecological Inference for Sets of R x C Contingency Tables*

### Description

This function (using the tuned parameters from Tune) fits a hierarchical model to ecological data in which the underlying contigency tables can have any number of rows or columns. The user supplies the data and may specify hyperprior values. Samples from the posterior distribution are returned as an mcmc object, which can be analyzed with functions in the coda package.

### Usage

```
Analyze(fstring, rho.vec, data = NULL, num.iters = 1e+06,
         save.every =1000, burnin = 10000,
         mu.vec.0 = rep(log((0.45/(mu.dim - 1))/0.55), mu.dim),
         kappa = 10, nu = (mu.dim + 6), psi = mu.dim,
         mu.vec.cu = runif(mu.dim, -3, 0), NNs.start = NULL,
         THETAS.start = NULL, prob.re = 0.15, sr.probs = NULL,
         sr.reps = NULL, keep.restart.info = FALSE,
         keepNNinternals = 0, keepTHETAS = 0, nolocalmode = 50,
         numscans = 1, Diri = 100, dof = 4, print.every = 10000,
         debug = 1)
```

## Arguments

`fstring` String: model formula of contingency tables' column totals versus row totals. Must be in specified format (an R character string and NOT a true R formula). See Details and Examples.

`rho.vec` Vector of dimension $I$ = number of contigency tables = number of rows in `data`: multipliers (usually in (0,1)) to the covariance matrix of the proposal distribution for the draws of the intermediate level parameters. Typically set to the vector output from `Tune`.

`data` Data frame.

`num.iters` Positive integer: The number of MCMC iterations for the sampler.

`save.every` Positive integer: The interval at which the draws will be saved. `num.iters` must be divisible by this value. Akin to `thin` in some packages. For example, `num.iters       = 1000` and `save.every = 10` outputs every 10th draw for a total of 100 outputed draws.

`burnin` Positive integer: The number of burn-in iterations for the sampler.

`mu.vec.0` Vector: mean of the (normal) hyperprior distribution for the $\mu$ parameter.

`kappa` Scalar: The diagonal of the covariance matrix for the (normal) hyperprior distribution for the $\mu$ parameter.

`nu` Scalar: The degrees of freedom for the (Inverse-Wishart) hyperprior distriution for the `SIGMA` parameter.

`psi` Scalar: The diagnoal of the matrix parameter of the (Inverse-Wishart) hyperprior distribution for the `SIGMA` parameter.

`mu.vec.cu` Vector of dimension $R * (C - 1)$, where $R(C)$ is the number of rows(columns) in each contigency table: Optional starting values for $\mu$ parameter.

`NNs.start` Matrix of dimension $I$ x $(R * C)$, where $I$ is the number of contingency tables = number of rows in `data`: Optional starting values for the internal cell counts, which must total to the continency table row and column totals contained in `data`. Use of the default (randomly generated internally) recommended.

`THETAS.start` Matrix of dimension $I$ x $(R * C)$, where $I$ is the number of contingency tables = number of rows in `data`: Optional starting values for the contingency table row probability vectors. The elements in each row of `THETAS.start` must meet $R$ sum-to-one constraints. Use of the default (randomly generated internally) recommended.

`prob.re` A positive fraction: Probability of random exchange in a parallel tempering fitting algorithm. Not yet implemented.

sr.probs
: Matrix of dimension $I$ x $R$: Each value represents the probability of selecting a particular contingency table's row as the row to be calculated deterministically in (product multinomial) proposals for Metropolis draws of the internal cell counts. For example, if R = 3 and row 2 of position `sr.probs` = c(.1, .5, .4), then in the third contingency table (correspoding to the third row of `data`), the proposal algorithm for the interior cell counts will calculate the third contingency table's first row deterministically with probability .1, the second row with probability .5, and the third row with probability .4. Use of default (generated internally) recommended.

sr.reps
: Matrix of dimension $I$ x $R$: Each value represents the number of times the (product multinomial proposal) Metropolis algorithm will be attempted when, in drawing the internal cell counts, the proposal for the corresponding contingency table row is to be calculated deterministically. sr.reps has the same structure as sr.probs, *i.e.*, position [3,1] of sr.reps corresponds to the third contingency table's first row. Use of default (generated internally) recommended.

keep.restart.info
: Logical: Whether last state of the chain should be saved to allow restart in the same state. Restart function not currently implemented.

keepNNinternals
: Positive integer: The number of draws of the internal cell counts in the contingency tables to be outputted. Must be divisible into `num.iters`. Use with caution: results in large RAM use even in modest-sized datasets.

keepTHETAS
: Positive integer: The number of draws of the contingency table row probability vectors in the contingency tables to be outputted. Must be divisible into `num.iters`. Use with caution: results in large RAM use even in modest-sized datasets.

nolocalmode
: Positive integer: How often an alternative drawing method for the contigency table internal cell counts will be used. Use of default value recommended.

numscans
: Positive integer: How often the algorithm to draw the contingency table internal cell counts will be implemented before new values of the other parameters are drawn. Use of default value recommended.

Diri
: Positive integer: How often a product Dirichlet proposal distribution will be used to draw the contingency table row probability vectors (the THETAS).

dof
: Positive integer: The degrees of freedom of the multivariate $t$ proposal distribution used in drawing the contingency table row probability vectors (the THETAS).

print.every
: Positive integer: If `debug == 1`, the number of every `print.every`$th$ iteration will be written to the screen. Must be divisible into `num.iters`.

debug
: Integer: Akin to `verbose` in some packages. If set to 1, certain status information (including rough notification regarding the number of iterations completed) will be written to the screen.

## Details

`Analyze` is the workhorse function in fitting the R x C ecological inference model described in Greiner & Quinn (2009).

Ecological data consist of sets of contingency tables in which the row and column totals, but none of the internal cell counts, are observed. For example, in the context of voting rights litigation, there is often one contigency table for each voting precinct; the row totals are voting-age population figures, with each row representing a race/ethnicity; all but the last (right-most) column representing votes cast for particular candidates; and the last (right-most) column representing persons not voting.

The model described in Greiner & Quinn (2009) conditions on the row totals throughout. In each contigency table, the rows are assumed to follow mutually independent multinomials, conditional on separate probability vectors which are denoted $theta_r$ for $r = 1$ to $R$ ($R$ being the number of rows in each contigency table). Each $theta_r$ then undergoes a multidimensional logistic transformation, using the last (right-most) column as the reference category. This results in $R$ transformed vectors of dimension $(C - 1)$; these transformed vectors, denoted $\omega_r s$, are stacked to form a single $\omega$ vector corresponding to that contingency table. The omega vectors are assumed to follow (i.i.d.) a multivariate normal distribution. A standard N($\mu_0$, $\kappa$ * I) and Inv-Wish($\nu$, $\psi$ * I) (I is the identity matrix) prior is placed on the normal. The user may set $\mu_0$, $\kappa$, $\nu$, amd $\psi$.

fstring must be in a specific format. It must be a string, and it must consist of (i) the names of vectors of contingency table column totals separated by commas, (ii) then a tilde, (iii) then the names of vectors of contingency table row totals separated by commas. The order in which the contigency table column totals are listed is important because the final column with become the reference category in the multidimensional logistic transformation described above. See Examples.

Fitting the model is accomplished via a Gibbs sampler in which the internal cell counts (for each contingency table), then the $\theta's$, and then the $\mu$ and $\Sigma$ parameters are drawn in turn. This method automatically produces draws of the internal cell counts, functions of which are often the true targets of inference.

The function returns an object of class mcmc suitable for use in functions from the coda package, including combination (with other outputs from Analyze) into an object of class mcmc.list. The return object includes draws from the posterior distribution of the following items: each element of the $\mu$; the standard deviations in $\Sigma$ (meaning the square root of the diagonal elements); the correlations in $\Sigma$; the sums across all contigency tables of the counts in each of the R * C internal cell positions; and a series of functions of these sums that are often of interest in voting applications (these may obviously be ignored if interest lies elsewhere). Except for the correlations from $\Sigma$, the labeling follows a self-evident pattern, with the names taking from `fstring`. The correlations are labeled by a combination of two numbers, representing their position in the $\Sigma$ matrix.

The series of functions of the internal cell counts calculated automatically fall into four categories: `LAMBDA`, `TURNOUT`, `GAMMA`, and `Beta`. To explain these terms, consider an example in which the contigency tables have three rows ("bla", "whi", and "his") and three columns ("Dem", "Rep", "Abs"), corresponding to black, white, Hispanic, Democratic, Republican, and Abstain (from voting). Thus, in position 1,1 of each contigency table is the (unobserved) number of blacks voting Democrat, position 2,1 holds the (unobserved) number of whites voting Democrat, etc. In each position (1,1 = black Democrat votes; 2,1 = white Democrat votes), sum across all $I$ contingency tables to produce a single R x C table consisting of summed counts (these sums are, incidentally, the NN values the software reports). `LAMBDA`, `TURNOUT`, `GAMMA`, and `Beta` are functions of these summed counts, as explained in the paragraph below. Note that the paragraph below refers to the counts in the single table produced by this summing process. Notation: $NN_{rc}$ is the sum (over all contingency tables) of the counts in cell r,c. So $NN_{bD}$ is the total number of blacks voting Democract, $NN_{wD}$ is the total number of whites voting Democrat, etc.

`LAMBDA`: For example, `LAMBDA_bD` = $NN_{bD}/(NN_b{-}NN_{bA})$. Similarly, `LAMBDA_hR` = $NN_{hR}/(NN_h - NN_{hA})$. In voting parlance, this the fraction of each race's *voters* supporting a particular candidate.

There are $R * (C - 1)$ LAMBDAs, $C - 1$ of them for each row.

`TURNOUT`: For example, $Turnout_w = (NN_w - NN_{wA})/NN_w$. In voting parlance, this is the fraction of each race that showed up to vote. There are R TURNOUTs, one for each row.

`GAMMA`: For example, `GAMMA_h` = $(NN_{hD} + NN_{hR})/(NN_{bD} + NN_{bR} + NN_{wD} + NN_{wR} + NN_{hD} + NN_{hR})$. In voting parlance, this is the fraction that each race contributes to the voting electorate.

`BETA`: For example, `BETA_wR` = $(NN_{wR})/(NN_{wD} + NN_{wR} + NN_{wA}) = (NN_{wR})/(NN_w)$. This is the fraction of each race's potential (as opposed to actual) voters supporting a particular candidate. Although there are theoretically $R * C$ `BETA` values that could be calculated, in fact the `BETA` values for the last (reference) category are ignored, so only $R * (C - 1)$ are calculated.

If `keepNNinternals` is non-zero, the specified number of draws of the internal cell counts for each contingency table will be save. These may be retreived via `attr` (see Examples, below). The result is a matrix of dimension `keepNNinternals` $\times R * C * I$, where $I$ is the number of contigency tables. Each row consists of an iteration's draws. The first column contains the draws of the counts in position 1,1 in the first contigency table, the second colum contains the draws of position 1,2 in the first contigency table, etc. In other words, the columns in the output represent the first contigency table vectorized row major, then the second contingency table vectorized row major, etc. The same applies to `keepTHETAS`, except that the `THETAs` represent the multinomial row probabilties.

## Value

An object of class `mcmc` suitable for use in functions in the coda package. Additional items, listed below, may be retrieved from this object, as detailed in the examples section.

`dim` Vector (integers) of length 2: number of saved simulations and number of automatically outputted parameters.

`dimnames` List: the first element `NULL` (currently not used), and the second element is a vector of the names of the automatically outputted parameters.

`acc.t` Vector of length $I$ = number of contigency tables: The fraction of multivariate $t$ proposals accepted in the Metropolis algorithm used to draw the `THETAs` (meaning the intermediate parameters in the hierarchy).

`acc.Diri` Vector of length $I$ = number of contigency tables: The fraction of Dirichlet-based proposals accepted in the Metropolis algorithm used to draw the `THETAs` (meaning the intermediate parameters in the hierarchy).

`vld.multinom` Matrix: To draw from the conditional posterior of the internal cell counts of a contigency table, the `Analyze` function draws R-1 vectors of lenth C from multinomial distributions. In then calculates the counts in the additional row (denote this row as r') deterministically. This procedure can result in negative values in row r', in which case the overall proposal for the interior cell counts is outside the parameter space (and thus invalid). vld.multinom keeps track of the percentage of proposals drawn in this manner that are valid (*i.e.*, not invalid). Each row of vld.multinom corresponds to a contingency table. Each column in vld.multinom corresponds to a *row* in the a contingency table. Each entry specifies the percentage of multinomial proposals that are valid when the specified contingency table row serves as the r' row. For instance, in position 5,2 of vld.multinom is the fraction of valid proposals for the 5th contingency table when the second contigency table row is the r'th row. A value of "NaN" means that `Analyze` chose to use a different (slower) method of drawing the internal cell counts because it suspected that the multinomial method would behave badly.

`acc.multinom` Matrix: Same as vld.multinom, except the entries represent the fraction of proposals accepted (instead of the fraction that are in the permissible parameter space).

`numrows.pt` Integer: Number of rows in each contingency table.

`numcols.pt` Integer: Number of columns in each contingency table.

`THETA` mcmc: Draws of the `THETA`s. See Details and Examples.

`NN.internals` mcmc: Draws of the internal cell counts. See Details and Examples.

## Warnings

**Computer time:** At present, using this function (and the others in this package) requires substantial computer time. The lack of information in ecological data results in slow mixing chains, and the number of parameters that must be drawn in each Gibbs sampler iteration is large. Chain length should be adjusted to achieve adequate convergence. In general, the more segregated the housing patterns in the jurisdiction (meaning the greater the percentage of contingency tables in which one row's counts make up a large portion of that table's total), the smaller the number of iterations needed. We are exploring more efficient sampling algorithms that we anticipate will result in better mixing and faster drawing. At present, however, users should anticipate that analysis of a dataset will take several hours.

**Large datasets:** At present, use of this fuction (and thus this package) is not recommended for large (*i.e.*, more than 1000 contingency tables) datasets. See immediately above.

**RAM requirements:** Do not select large values of `keepNNinternals` or `keepTHETAS` without adequate RAM.

**Gelman-Rubin diagnostic in the CODA package:** Using the Gelman-Rubin convergence diagnostic as presently implemented in the CODA package (called by `gelman.diag()`) on multiple chains produced by Analyze will cause an error. The reason is that some of the NN.internals and functions of them ($LAMBDAs$, `TURNOUT`s, $GAMMAs$, and $BETAs$) are linearly dependant, and the current coda implmentation of gelman.diag() relies on a matrix inversion.

## Author(s)

D. James Greiner, Paul D. Baines, \& Kevin M. Quinn

## References

D. James Greiner \& Kevin M. Quinn. 2009. "R x C Ecological Inference: Bounds, Correlations, Flexibility, and Transparency of Assumptions." *J.R. Statist. Soc. A* 172:67-81.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2002. *Output Analysis and Diagnostics for MCMC (CODA).*

## Examples

```
## Not run:
library(RxCEcolInf)
data(stlouis)
Tune.stlouis <- Tune("Bosley, Roberts, Ribaudo, Villa, NoVote ~ bvap, ovap",
                     data = stlouis,
                     num.iters = 10000,
                     num.runs = 15)
Chain1.stlouis <- Analyze("Bosley, Roberts, Ribaudo, Villa,
                          NoVote ~ bvap, ovap",
                          rho.vec = Tune.stlouis$rhos,
                          data = stlouis,
                          num.iters = 1500000,
                          burnin = 150000,
                          save.every = 1500,
                          print_every = 15000,
                          debug = 1,
                          keepNNinternals = 100,
                          keepTHETAS = 100)
Chain2.stlouis <- Analyze("Bosley, Roberts , Ribaudo, Villa,
                          NoVote ~ bvap, ovap",
                          rho.vec = Tune.stlouis$rhos,
                          data = stlouis,
                          num.iters = 1500000,
                          burnin = 150000,
                          save.every = 1500,
                          print_every = 15000,
                          debug = 1,
                          keepNNinternals = 100,
                          keepTHETAS = 100)
Chain3.stlouis <- Analyze("Bosley, Roberts , Ribaudo, Villa,
                          NoVote ~ bvap, ovap",
                          rho.vec = Tune.stlouis$rhos,
                          data = stlouis,
                          num.iters = 1500000,
                          burnin = 150000,
                          save.every = 1500,
                          print_every = 15000,
                          debug = 1,
                          keepNNinternals = 100,
                          keepTHETAS = 100)
stlouis.MCMClist <- mcmc.list(Chain1.stlouis, Chain2.stlouis,
Chain3.stlouis)
names(attributes(stlouis.MCMClist))
summary(stlouis.MCMClist, quantiles = c(.025, .05, .5, .95, .975))
plot(stlouis.MCMClist)
geweke.diag(stlouis.MCMClist)
heidel.diag(stlouis.MCMClist)
#  Do not run gelman.diag; see warnings
NNs <- attr(stlouis.MCMClist, "NN.internals")
THETAS <- attr(stlouis.MCMClist, "THETA")
```

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

## AnalyzeWithExitPoll *Workhorse Function for Ecological Inference for Sets of R x C Contingency Tables When Incorporating a Survey such as an Exit Poll*

### Description

This function (using the tuned parameters from `TuneWithExitPoll`) fits a hierarchical model to data from two sources: (i) ecological data and in which the underlying contigency tables can have any number of rows or columns, and (ii) data from a survey sample of some of the contingency tables. The user supplies the data and may specify hyperprior values. Samples from the posterior distribution are returned as an `mcmc` object, which can be analyzed with functions in the `coda` package.

### Usage

```
AnalyzeWithExitPoll(fstring, rho.vec, exitpoll, data = NULL,
            num.iters = 1e+06, save.every = 1000, burnin = 10000,
                   mu.vec.0 = rep(log((0.45/(mu.dim - 1))/0.55), mu.dim),
                   kappa = 10, nu = (mu.dim + 6), psi = mu.dim,
                   mu.vec.cu = runif(mu.dim, -3, 0), NNs.start = NULL,
                   MMs.start = NULL, THETAS.start = NULL, sr.probs = NULL,
                   sr.reps = NULL, keep.restart.info = FALSE,
                   keepNNinternals = 0, keepTHETAS = 0, nolocalmode = 50,
                   numscans = 1, Diri = 100, dof = 4, eschew = FALSE,
                   print.every = 10000, debug = 1)
```

### Arguments

`fstring` String: model formula of contingency tables' column totals versus row totals. Must be in specified format (an R character string and NOT a true R formula). See Details and Examples.

`rho.vec` Vector of dimension $I$ = number of contigency tables = number of rows in `data`: multipliers (usually in (0,1)) to the covariance matrix of the proposal distribution for the draws of the intermediate level parameters. Typically set to the vector output from `TuneWithExitPoll`.

`exitpoll` Matrix of dimensions $I$ = number of contingency tables = number of rows in `data` by (R * C) = number of cells in each contingency table: The results of a survey sample of some of the contingency tables. Must be in specified format. See Details.

`data` Data frame.

`num.iters` Positive integer: The number of MCMC iterations for the sampler.

`save.every` Positive integer: The interval at which the draws will be saved. `num.iters` must be divisible by this value. Akin to `thin` in some packages. For example, `num.iters = 1000` and `save.every = 10` outputs every 10th draw for a total of 100 outputed draws.

`burnin` Positive integer: The number of burn-in iterations for the sampler.

`mu.vec.0` Vector: mean of the (normal) hyperprior distribution for the $\mu$ parameter.

`kappa` Scalar: The diagonal of the covariance matrix for the (normal) hyperprior distribution for the $\mu$ parameter.

`nu` Scalar: The degrees of freedom for the (Inverse-Wishart) hyperprior distriution for the `SIGMA` parameter.

`psi` Scalar: The diagnoal of the matrix parameter of the (Inverse-Wishart) hyperprior distribution for the `SIGMA` parameter.

`mu.vec.cu` Vector of dimension $R * (C - 1)$, where $R(C)$ is the number of rows(columns) in each contigency table: Optional starting values for $\mu$ parameter.

`NNs.start` Matrix of dimension $I$ x $(R * C)$, where $I$ is the number of contingency tables = number of rows in `data`: Optional starting values for the internal cell counts, which must total to the continency table row and column totals contained in `data`. Use of the default (randomly generated internally) recommended.

`MMs.start` Matrix of dimension $I$ x $(R * C)$, where $I$ is the number of contingency tables = number of rows in `data`: Optional starting values for the *missing* internal cell counts, which must total to the continency table row and column totals contained in `data`. By *missing* internal cell counts we mean the counts in the contigency tables not observed in the survey sample or `exitpoll`. Use of the default (randomly generated internally) recommended.

`THETAS.start` Matrix of dimension $I$ x $(R * C)$, where $I$ is the number of contingency tables = number of rows in `data`: Optional starting values for the contingency table row probability vectors. The elements in each row of `THETAS.start` must meet $R$ sum-to-one constraints. Use of the default (randomly generated internally) recommended.

`sr.probs` Matrix of dimension $I$ x $R$: Each value represents the probability of selecting a particular contingency table's row as the row to be calculated deterministically in (product multinomial) proposals for Metropolis draws of the internal cell counts. For example, if R = 3 and row 2 of position `sr.probs` = c(.1, .5, .4), then in the third contingency table (correspoding to the third row of `data`), the proposal algorithm for the interior cell counts will calculate the third contingency table's first row deterministically with probability .1, the second row with probability .5, and the third row with probability .4. Use of default (generated internally) recommended.

`sr.reps` Matrix of dimension $I$ x $R$: Each value represents the number of times the (product multinomial proposal) Metropolis algorithm will be attempted when, in drawing the internal cell counts, the proposal for the corresponding contingency table row is to be calculated deterministically. sr.reps has the same structure as sr.probs, *i.e.*, position [3,1] of sr.reps corresponds to the third contingency table's first row. Use of default (generated internally) recommended.

`keep.restart.info`
Logical: Whether last state of the chain should be saved to allow restart in the same state. Restart function not currently implemented.

`keepNNinternals`
Positive integer: The number of draws of the internal cell counts in the contingency tables to be outputted. Must be divisible into `num.iters`. Use with caution: results in large RAM use even in modest-sized datasets.

`keepTHETAS` Positive integer: The number of draws of the contingency table row probability vectors in the contingency tables to be outputted. Must be divisible into `num.iters`. Use with caution: results in large RAM use even in modest-sized datasets.

`nolocalmode` Positive integer: How often an alternative drawing method for the contigency table internal cell counts will be used. Use of default value recommended.

`numscans` Positive integer: How often the algorithm to draw the contingency table internal cell counts will be implemented before new values of the other parameters are drawn. Use of default value recommended.

`Diri` Positive integer: How often a product Dirichlet proposal distribution will be used to draw the contingency table row probability vectors (the THETAS).

`dof` Positive integer: The degrees of freedom of the multivariate $t$ proposal distribution used in drawing the contingency table row probability vectors (the THETAS).

`eschew` Logical: If true, calculation of certain functions of the interntal cell counts omits the two right-most columns instead of only the single right-most column. Not yet implemented.

`print.every` Positive integer: If `debug == 1`, the number of every `print.every`$th$ iteration will be written to the screen. Must be divisible into `num.iters`.

`debug` Integer: Akin to `verbose` in some packages. If set to 1, certain status information (including rough notification regarding the number of iterations completed) will be written to the screen.

## Details

`AnalyzeWithExitPoll` is the workhorse function in fitting the R x C ecological inference model described in Greiner & Quinn (2009) with the addition of information from a survey sample from some of the contingency tables. Details and terminology of the basic (*i.e.*, without a survey sample) data structure and ecological inference model are discussed in the documentation accompanying the `Analyze` function.

In the present implementation, the `AnalyzeWithExitPoll` presumes that the survey consisted of a simple random sample from the in-sample contingency tables. Future implementations will allow incorporation of more complicated survey sampling schemes.

The arguments to `AnalyzeWithExitPoll` are essentially identical to those of `Analyze` with the major exception of `exitpoll`. `exitpoll` feeds the results of the survey sample to the function, and a particular format is required. Specifically, `exitpoll` must have the same number of rows as `data`, meaning one row for each contigency table in the dataset. It must have R * C columns, meaning one column for each cell in one of the ecological data's contingency tables. The first row of `exitpoll` must correspond to the first row of `data`, meaning that the two rows must contain information from the same contingency table. The second row of `exitpoll` must contain information from the contingency table represented in the second row of `data`. And so on. Finally, `exitpoll` must have counts from the sample of the contingency table in vectorized row major format.

To illustrate with a voting example: Suppose the contingency tables have two rows, labeled bla and whi, and three columns, denoted Dem, Rep, and Abs. In other words, the `fstring` argument would be `"Dem, Rep, Abs ~  bla, whi"`. Suppose there are 100 contingency tables. The `data` will be of dimension $100 \times 5$, with each row consisting of the row and column totals from that particular contigency table. `exitpoll` will be of dimension $100 \times 6$. Row 11 of the `exitpoll` will consist of the following: in position 1, the number of blacks voting Democrat observed in the sample of contingency table 11; in position 2, the number of blacks voting Republican observed in the sample of contigency table 11; in position 3, the number of blacks Abstaining from voting observed in the sample of contingency table 11; in position 4, the number of whites voting Democrat observed in the sample of contingency table 11; etc.

For tables in which there was no sample taken (*i.e.*, out-of-sample tables), the corresponding row of `exitpoll` should have a vector of 0s.

Model fitting proceeds similarly as in `Analyze`, and output is simimilarly similar. See documentation accompanyng `Analyze` for further information.

## Value

An object of class `mcmc` suitable for use in functions in the coda package. Additional items, listed below, may be retrieved from this object, as detailed in the examples section.

`dim` Vector (integers) of length 2: number of saved simulations and number of automatically outputted parameters.

`dimnames` List: the first element `NULL` (currently not used), and the second element is a vector of the names of the automatically outputted parameters.

`acc.t` Vector of length $I$ = number of contigency tables: The fraction of multivariate $t$ proposals accepted in the Metropolis algorithm used to draw the `THETA`s (meaning the intermediate parameters in the hierarchy).

`acc.Diri` Vector of length $I$ = number of contigency tables: The fraction of Dirichlet-based proposals accepted in the Metropolis algorithm used to draw the `THETA`s (meaning the intermediate parameters in the hierarchy).

`vld.multinom` Matrix: To draw from the conditional posterior of the internal cell counts of a contigency table, the `Analyze` function draws R-1 vectors of lenth C from multinomial distributions. In then calculates the counts in the additional row (denote this row as r') deterministically. This procedure can result in negative values in row r', in which case the overall proposal for the interior cell counts is outside the parameter space (and thus invalid). vld.multinom keeps track of the percentage of proposals drawn in this manner that are valid (*i.e.*, not invalid). Each row of vld.multinom corresponds to a contingency table. Each column in vld.multinom corresponds to a *row* in the a contingency table. Each entry specifies the percentage of multinomial proposals that are valid when the specified contingency table row serves as the r' row. For instance, in position 5,2 of vld.multinom is the fraction of valid proposals for the 5th contingency table when the second contigency table row is the r'th row. A value of “NaN” means that `Analyze` chose to use a different (slower) method of drawing the internal cell counts because it suspected that the multinomial method would behave badly.

`acc.multinom` Matrix: Same as vld.multinom, except the entries represent the fraction of proposals accepted (instead of the fraction that are in the permissible parameter space).

`numrows.pt` Integer: Number of rows in each contingency table.

`numcols.pt` Integer: Number of columns in each contingency table.

`THETA` mcmc: Draws of the `THETA`s. See Details and Examples.

`NN.internals` mcmc: Draws of the internal cell counts. See Details and Examples.

## Warnings

**Computer time:** At present, using this function (and the others in this package) requires substantial computer time. The lack of information in ecological data results in slow mixing chains, and the number of parameters that must be drawn in each Gibbs sampler iteration is large. Chain length should be adjusted to achieve adequate convergence. In general, the more segregated the housing patterns in the jurisdiction (meaning the greater the percentage of contingency tables in which one row's counts make up a large portion of that table's total), the smaller the number of iterations needed. We are exploring more efficient sampling algorithms that we anticipate will result in better mixing and faster drawing. At present, however, users should anticipate that analysis of a dataset will take several hours.

**Large datasets:** At present, use of this fuction (and thus this package) is not recommended for large (*i.e.*, more than 1000 contingency tables) datasets. See immediately above.

**RAM requirements:** Do not select large values of `keepNNinternals` or `keepTHETAS` without adequate RAM.

**Gelman-Rubin diagnostic in the coda package:** Using the Gelman-Rubin convergence diagnostic as presently implemented in the CODA package (called by `gelman.diag()`) on multiple chains produced by Analyze will cause an error. The reason is that some of the NN.internals and functions of them ($LAMBDAs$, `TURNOUT`s, $GAMMAs$, and $BETAs$) are linearly dependant, and the current coda implmentation of gelman.diag() relies on a matrix inversion.

## Author(s)

D. James Greiner, Paul D. Baines, \& Kevin M. Quinn

## References

D. James Greiner \& Kevin M. Quinn. 2009. "R x C Ecological Inference: Bounds, Correlations, Flexibility, and Transparency of Assumptions." *J.R. Statist. Soc. A* 172:67-81.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2002. *Output Analysis and Diagnostics for MCMC (CODA).*

## Examples

```
## Not run:
SimData <- gendata.ep()    #  simulated data
FormulaString <- "Dem, Rep, Abs ~ bla, whi, his"
EPInvTune <-  TuneWithExitPoll(fstring = FormulaString,
                              data = SimData$GQdata,
                              exitpoll=SimData$EPInv$returnmat.ep,
                              num.iters = 10000,
                              num.runs = 15)
EPInvChain1 <- AnalyzeWithExitPoll(fstring = FormulaString,
                                  data = SimData$GQdata,
                                  exitpoll=SimData$EPInv$returnmat.ep,
                                  num.iters = 2000000,
                                  burnin = 200000,
                                  save.every = 2000,
                                  rho.vec = EPInvTune$rhos,
                                  print.every = 20000,
                                  debug = 1,
                                  keepTHETAS = 0,
                                  keepNNinternals = 0)
EPInvChain2 <- AnalyzeWithExitPoll(fstring = FormulaString,
                                  data = SimData$GQdata,
                                  exitpoll=SimData$EPInv$returnmat.ep,
                                  num.iters = 2000000,
                                  burnin = 200000,
                                  save.every = 2000,
                                  rho.vec = EPInvTune$rhos,
                                  print.every = 20000,
                                  debug = 1,
                                  keepTHETAS = 0,
                                  keepNNinternals = 0)
EPInvChain3 <- AnalyzeWithExitPoll(fstring = FormulaString,
                                  data = SimData$GQdata,
                                  exitpoll=SimData$EPInv$returnmat.ep,
                                  num.iters = 2000000,
                                  burnin = 200000,
                                  save.every = 2000,
                                  rho.vec = EPInvTune$rhos,
                                  print.every = 20000,
                                  debug = 1,
                                  keepTHETAS = 0,
                                  keepNNinternals = 0)
EPInv <- mcmc.list(EPInvChain1, EPInvChain2, EPInvChain3)

## End(Not run)
```

## gendata.ep — *Function To Simulate Ecological and Survey Data For Use in Testing And Analyzing Other Functions in Package*

### Description

This function generates simulated ecological data, *i.e.*, data in the form of contigency tables in which the row and column totals but none of the internal cell counts are observed. At the user's option, data from simulated surveys of some of the 'units' (in voting parlance, 'precincts') that gave rise to the contingency tables are also produced.

### Usage

```
gendata.ep(nprecincts = 175,
              nrowcat = 3,
              ncolcat = 3,
              colcatnames = c("Dem", "Rep", "Abs"),
              mu0 = c(-.6, -2.05, -1.7, -.2, -1.45, -1.45),
              rowcatnames = c("bla", "whi", "his", "asi"),
              alpha = c(.35, .45, .2, .1),
              housing.seg = 1,
              nprecincts.ep = 40,
              samplefrac.ep = 1/14,
              K0 = NULL,
              nu0 = 12,
              Psi0 = NULL,
              lambda = 1000,
              dispersion.low.lim = 1,
              dispersion.up.lim = 1,
              outfile=NULL,
              his.agg.bias.vec = c(0,0),
              HerfInvexp = 3.5,
              HerfNoInvexp = 3.5,
              HerfReasexp = 2)
```

## Arguments

`nprecincts` positive integer: The number of contingency tables (precincts) in the simulated dataset.

`nrowcat` integer > 1: The number of rows in each of the contingency tables.

`ncolcat` integer > 1: The number of columns in each of the contingency tables.

`rowcatnames` string of length = length(`nrowcat`): Names of rows in each contingency table.

`colcatnames` string of length = length(`ncolcat`): Names of columns in each contingency table.

`alpha` vector of length(`nrowcat`): initial parameters to a Dirichlet distribution used to generate each contingency table's row fractions.

`housing.seg` scalar > 0: multiplied to alpha to generate final parameters to Dirichlet distribution used to generate each contingency table's row fractions.

`mu0` vector of length (`nrowcat` * (`ncolcat` - 1)): The mean of the multivariate normal hyperprior at the top level of the hierarchical model from which the data are simulated. See Details.

`K0` square matrix of dimension (`nrowcat` * (`ncolcat` - 1)): the covariance matrix of the multivariate normal hyperprior at the top level of the hierarchical model from which the data are simulated. See Details.

`nu0` scalar > 0: the degrees of freedom for the Inv-Wishart hyperprior from which the $SIGMA$ matrix will be drawn.

`Psi0` square matrix of dimension (`nrowcat` * (`ncolcat` - 1)): scale matrix for the Inv-Wishart hyperprior from which the `SIGMA` matrix will be drawn.

`lambda` scalar > 0: initial parameter of the Poisson distribution from which the number of voters in each precinct will be drawn

`dispersion.low.lim`
: scalar > 0 but < dispersion.up.lim: lower limit of a draw from `runif()` to be multiplied to `lambda` to set a lower limit on the parameter used to draw from the Poisson distribution that determines the number of voters in each precinct.

`dispersion.up.lim`
: scalar > dispersion.low.lim: upper limit of a draw from `runif()` to be multiplied to `lambda` to set a upper limit on the parameter used to draw from the Poisson distribution that determines the number of voters in each precinct.

`outfile`
: string ending in ".Rdata": filepath and name of object; if non-NULL, the object returned by this function will be saved to the location specified by `outfile`.

`his.agg.bias.vec`
: vector of length 2: only implemented for nowcat = 3 and ncolcat = 3: if non-null, induces aggregation bias into the simulated data. See Details.

`nprecincts.ep`
: integer > -1 and less than nprecincts: number of contingency tables (precincts) to be included in simulated survey sample (ep for "exit poll").

`samplefrac.ep`
: fraction (real number between 0 and 1): percentage of individual units (voters) within each contingency table (precinct) include in the survey sample.

`HerfInvexp`
: scalar: exponent used to generate inverted quasi-Herfindahl weights used to sample contingency tables (precincts) for inclusion in a sample survey. See Details.

`HerfNoInvexp`
: scalar: same as HerInvexp except the quasi-Herfindahl weights are not inverted. See Details.

`HerfReasexp`
: scalar: same as HerfInvexp, for a separate sample survey. See Details.

## Details

This function simulates data from the ecological inference model outlined in Greiner \& Quinn (2009). At the user's option (by setting nprecincts.ep to an integer greater than 0), the function generates three survey samples from the simulated dataset. The specifics of the function's operation are as follows.

First, the function simulates the total number of individual units (voters) in each contigency table (precinct) from a Poisson distribution with parameter `lambda` * runif(1, dispersion.low.lim, dispersion.up.lim). Next, for each table, the function simulates the vector of fraction of units (voters) in each table (precinct) row. The fractions are simulated from a Dirichlet distribution with parameter vector `housing.seg` * `alpha`. The row fractions are multiplied by the total number of units (voters), and the resulting vector is rounded to produce contingency table row counts for each table.

Next, a vector $mu$ is simulated from a multivariate normal with mean `mu0` and covariance matrix `K0`. A covariance matrix `Sigma` is simulated from an Inv-Wishart with `nu0` degrees of freedom and scale matrix `Psi0`.

Next, `nprecincts` vectors are drawn from $N(mu, SIGMA)$. Each of these draws undergoes an inverse-stacked multidimensional logistic transformation to produce a set of `nrowcat` probability vectors (each of which sums to one) for `nrowcat` multinomial distributions, one for each row in that contingency table. Next, the `nrowcat` multinomial values, which represent the true (and in real life, unobserved) internal cell counts, are drawn from the relevant row counts and these probability vectors. The column totals are calculated via summation.

If `nprecincts.ep` is greater than 0, three simulated surveys (exit polls) are drawn. All three select contingency tables (precincts) using weights that are a function of the composition of the row totals. Specifically the row fractions are raised to a power q and then summed (when q = 2 this calculation is known in antitrust law as a Herfindahl index). For one of the three surveys (exit polls) `gendata.ep` generates, these quasi-Herfindahl indices are the weights. For two of the three surveys (exit polls) `gendata.ep` generates, denoted `EPInv` and `EPReas`, the sample weights are the reciprocals of these quasi-Herfindhal indices. The former method tends to weight contingency tables (precincts) in which one row dominates the table higher than contigency tables (precincts) in which row fractions are close to the same. In voting parlance, precincts in which one racial group dominates are more likely to be sampled than racially mixed precincts. The latter method, in which the sample weights are reciprocated, weights contingency tables in which row fractions are similar more highly; in voting parlance, mixed-race precincts are more likly to be sampled.

For example, suppose `nrowcat` = 3, `HerInvexp` = 3.5, `HerfReas` = 2, and `HerfNoInv` = 3.5. Consider contingency table P1 with row counts (300, 300, 300) and contingency table P2 with row counts (950, 25, 25). Then:

**Row fractions:** The corresponding row fractions are (300/900, 300/900, 300/900) = (.33, .33, .33) and (950/1000, 25/1000, 25/1000) = (.95, .025, .025).

**EPInv weights:** `EPInv` would sample from assign P1 and P2 weights as follows: $1/sum(.33^3.5, .33^3.5, .33^3.5) = 16.1$ and $1/sum(.95^3.5, .025^3.5, .025^3.5) = 1.2$.

**EPReas weights:** `EPReas` would assign weights as follows: $1/sum(.33^2, .33^2, .33^2) = 3.1$ and $1/sum(.95^2, .025^2, .025^2) = 1.1$.

**EPNoInv weights:** `EPNoInv` would assign weights as follows: $sum(.33^3.5, .33^3.5, .33^3.5) = .062$ and $sum(.95^3.5, .025^3.5, .025^3.5) = .84$.

For each of the three simulated surveys (`EPInv`, `EPReas`, and `EPNoInv`), `gendata.ep` returns a list of length three. The first element of the list, `returnmat.ep`, is a matrix of dimension `nprecincts` by (`nrowcat` * `ncolcat`) suitable for passing to `TuneWithExitPoll` and `AnalyzeWithExitPoll`. That is, the first row of `returnmat.ep` corresponds to the first row of `GQdata`, meaning that they both contain information from the same contingency table. The second row of `returnmat.ep` contains information from the contingency table represented in the second row of `GQdata`. And so on. In addition, `returnmat.ep` has counts from the sample of the contingency table in vectorized row major format, as required for `TuneWithExitPoll` and `AnalyzeWithExitPoll`.

If `nrowcat` = `ncolcat` = 3, then the user may set `his.agg.bias.vec` to be nonzero. This will introduce aggregation bias into the data by making the probability vector of the second row of each contingency table a function of the fractional composition of the third row. In voting parlance, if the rows are black, white, and Hispanic, the white voting behavior will be a function of the percent Hispanic in each precinct. For example, if `his.agg.bias.vec` = c(1.7, -3), and if the fraction Hispanic in each precinct i is $X_hi$, then in the ith precinct, the $mu_i[3]$ is set to `mu0[3]` + $X_hi * 1.7$, while $mu_i[4]$ is set to `mu0[4]` + $X_hi * -3$. This feature allows testing of the ecological inference model with aggregation bias.

## Value

A list with the follwing elements.

`GQdata` Matrix of dimension `nprecincts` by (`nrowcat` + `ncolcat`): The simulated (observed) ecological data, meaning the row and column totals in the contingency tables. May be passed as `data` argument in `Tune`, `Analyze`, `TuneWithExitPoll`, and `AnalyzeWithExitPoll`

`EPInv` List of length 3: `returnmat.ep`, the first element in the list, is a matrix that may be passed as the `exitpoll` argument in `TuneWithExitPoll` and `AnalyzeWithExitPoll`. See Details. `ObsData` is a dataframe that may be used as the `data` argument in the `survey` package. `sampprecincts.ep` is a vector detailing the row numbers of `GQdata` (meaning the contingency tables) that were included in the `EPInv` survey (exit poll). See Details for an explanation of the weights used to select the contingency tables for inclusion in the `EPInv` survey (exit poll).

`EPNoInv` List of length 3: Contains the same elements as `EPInv`. See Details for an explanation of weights used to select the contingency tables for inclusion in the `EPNoInv` survey (exit poll).

`EPReas` List of length 3: Contains the same elements as `EPInv`. See Details for an explanation of weights used to select the contingency tables for inclusion in the `EPReas` survey (exit poll).

`omega.matrix` Matrix of dimension `nprecincts` by (`nrowcat` * (`ncolcat`-1)): The matrix of draws from the multivariate normal distribution at the second level of the hiearchical model giving rise to `GQdata`. These values undergo an inverse-stacked-multidimensional logistic transformation to produce contingency table row probability vectors.

`interior.tables`

List of length `nprecincts`: Each element of the list is a full (meaning all interior cells are filled in) contingency table.

`mu` vector of length `nrowcat` * (`ncolcat`-1): the $mu$ vector drawn at the top level of the hierarchical model giving rise to `GQdata`. See Details.

`Sigma` square matrix of dimension `nrowcat` * (`ncolcat`-1): the covariance matrix drawn at the top level of the hierarchical model giving rise to `GQdata`. See Details.

`Sigma.diag` the output of `diag(Sigma)`.

`Sigma.corr` the output of `cov2cor(Sigma)`.

`sim.check.vec` vector: the true values of the parameters generated by `Analyze` and `AnalyzeWithExitPoll` in the same order as the parameters are produced by those two functions. This vector is useful in assessing the coverage of intervals from the posterior draws from `Analyze` and `AnalyzeWithExitPoll`.

## Author(s)

D. James Greiner \& Kevin M. Quinn

## References

D. James Greiner \& Kevin M. Quinn. 2009. "R x C Ecological Inference: Bounds, Correlations, Flexibility, and Transparency of Assumptions." *J.R. Statist. Soc. A* 172:67-81.

## Examples

```
## Not run:
SimData <- gendata.ep()    #  simulated data
FormulaString <- "Dem, Rep, Abs ~ bla, whi, his"
EPInvTune <-  TuneWithExitPoll(fstring = FormulaString,
                               data = SimData$GQdata,
                               exitpoll=SimData$EPInv$returnmat.ep,
                               num.iters = 10000,
                               num.runs = 15)
EPInvChain1 <- AnalyzeWithExitPoll(fstring = FormulaString,
                                   data = SimData$GQdata,
                                   exitpoll=SimData$EPInv$returnmat.ep,
                                   num.iters = 2000000,
                                   burnin = 200000,
                                   save.every = 2000,
                                   rho.vec = EPInvTune$rhos,
                                   print.every = 20000,
                                   debug = 1,
                                   keepTHETAS = 0,
                                   keepNNinternals = 0)
EPInvChain2 <- AnalyzeWithExitPoll(fstring = FormulaString,
                                   data = SimData$GQdata,
                                   exitpoll=SimData$EPInv$returnmat.ep,
                                   num.iters = 2000000,
                                   burnin = 200000,
                                   save.every = 2000,
                                   rho.vec = EPInvTune$rhos,
                                   print.every = 20000,
                                   debug = 1,
                                   keepTHETAS = 0,
                                   keepNNinternals = 0)
EPInvChain3 <- AnalyzeWithExitPoll(fstring = FormulaString,
                                   data = SimData$GQdata,
                                   exitpoll=SimData$EPInv$returnmat.ep,
                                   num.iters = 2000000,
                                   burnin = 200000,
                                   save.every = 2000,
                                   rho.vec = EPInvTune$rhos,
                                   print.every = 20000,
                                   debug = 1,
                                   keepTHETAS = 0,
                                   keepNNinternals = 0)
EPInv <- mcmc.list(EPInvChain1, EPInvChain2, EPInvChain3)

## End(Not run)
```

## stlouis *Precinct-Level Data from 1993 St. Louis Mayoral Election*

## Description

Precinct-Level Data from 1993 St. Louis Mayoral Election

## Usage

```
data(stlouis)
```

## Format

A data frame with 349 observations on the following 7 variables. Each observation is a precinct.

`bvap` Black voting age population

`ovap` Non-black (other) voting age population

`Bosley` Votes for Bosley

`Roberts` Votes for Roberts

`Ribaudo` Votes for Ribaudo

`Villa` Votes for Villa

`NoVote` Number of non-voters

## Source

Personal communication with Brady Baybeck.

---

# `texas` *Precinct-level Data, Bush v. Gore, Tex Cong Dist 24*

---

## Description

Precinct-level data for Bush versus Gore in the precincts that made up Texas Congressional District 24 as it existed in 2000, using Census 2000 data.

## Usage

```
data(texas)
```

## Format

A data frame with 249 observations on the following 6 variables. Each observation is a precinct.

`black` Black voting age population

`white` White voting age population

`hispanic` Hispanic voting age population

`Democrat` Votes for Democratic candidate

`Republican` Votes for Republican candidate

`Abstain` Number of voting age citizens who did not vote

### Source

Lublin and Voss, Federal Elections Project, American University, http://spa.american.edu/ccps/pages.php?ID=12

## Tune — *Tuning Function for Ecological Inference for Sets of R x C Contingency Tables*

### Description

This function tunes the markov chain monte carlo algorithm used to fit a hierarchical model to ecological data in which the underlying contigency tables can have any number of rows or columns. The user supplies the data and may specify hyperprior values. The function's primary output is a vector of multipliers, called `rhos`, used to adjust the covariance matrix of the multivariate $t_4$ distribution used to propose new values of intermediate-level parameters (denoted THETAS).

### Usage

```
Tune(fstring, data=NULL, num.runs=12, num.iters=10000,
     rho.vec=rep(0.05, ntables),
     kappa=10, nu=(mu.dim+6), psi=mu.dim,
     mu.vec.0=rep(log((.45/(mu.dim-1))/.55), mu.dim),
     mu.vec.cu=runif(mu.dim, -3, 0),
     nolocalmode=50, sr.probs=NULL, sr.reps=NULL,
     numscans=1, Diri=100, dof=4, debug=1)
```

### Arguments

`fstring` String: model formula of contingency tables' column totals versus row totals. Must be in specified format (an R character string and NOT a true R formula). See Details and Examples.

`data` Data frame.

`num.runs` Positive integer: The number of runs or times (each of `num.iters` iterations) the tuning algorthm will be implemented.

`num.iters` Positive integer: The number of iterations in each run of the tuning algorithm.

`rho.vec` Vector of dimension $I$ = number of contigency tables = number of rows in `data`: initial values of multipliers (usually in (0,1)) to the covariance matrix of the proposal distribution for the draws of the intermediate level parameters. The purpose of this `Tune` function is to adjust these values so as to achieve acceptance ratios of between .2 and .5 in the MCMC draws of the `THETA`s.

`kappa` Scalar: The diagonal of the covariance matrix for the (normal) hyperprior distribution for the $\mu$ parameter.

`nu` Scalar: The degrees of freedom for the (Inverse-Wishart) hyperprior distriution for the `SIGMA` parameter.

`psi` Scalar: The diagonal of the matrix parameter of the (Inverse-Wishart) hyperprior distribution for the `SIGMA` parameter.

`mu.vec.0` Vector: mean of the (normal) hyperprior distribution for the $\mu$ parameter.

`mu.vec.cu` Vector of dimension $R * (C - 1)$, where $R(C)$ is the number of rows(columns) in each contigency table: Optional starting values for $\mu$ parameter.

`nolocalmode` Positive integer: How often an alternative drawing method for the contigency table internal cell counts will be used. Use of default value recommended.

`sr.probs` Matrix of dimension $I$ x $R$: Each value represents the probability of selecting a particular contingency table's row as the row to be calculated deterministically in (product multinomial) proposals for Metropolis draws of the internal cell counts. For example, if R = 3 and row 2 of position `sr.probs` = c(.1, .5, .4), then in the third contingency table (correspoding to the third row of `data`), the proposal algorithm for the interior cell counts will calculate the third contingency table's first row deterministically with probability .1, the second row with probability .5, and the third row with probability .4. Use of default (generated internally) recommended.

`sr.reps` Matrix of dimension $I$ x $R$: Each value represents the number of times the (product multinomial proposal) Metropolis algorithm will be attempted when, in drawing the internal cell counts, the proposal for the corresponding contingency table row is to be calculated deterministically. sr.reps has the same structure as sr.probs, *i.e.*, position [3,1] of sr.reps corresponds to the third contingency table's first row. Use of default (generated internally) recommended.

`numscans` Positive integer: How often the algorithm to draw the contingency table internal cell counts will be implemented before new values of the other parameters are drawn. Use of default value recommended.

`Diri` Positive integer: How often a product Dirichlet proposal distribution will be used to draw the contingency table row probability vectors (the THETAS).

`dof` Positive integer: The degrees of freedom of the multivariate $t$ proposal distribution used in drawing the contingency table row probability vectors (the THETAS).

`debug` Integer: Akin to `verbose` in some packages. If set to 1, certain status information (including rough notification regarding the number of iterations completed) will be written to the screen.

## Details

Tune is a necessary precursor function to `Analyze`, the workhorse function in fitting the R x C ecological inference model described in Greiner & Quinn (2009). The details of this model are discussed in the documentation accompanying `Analyze`.

One of the stages of the Gibbs sampler used to fit the Greiner & Quinn ecological inference model involves sampling from the conditional posterior distribution of the vector of probabilities associated with each contingency table (precinct, in voting applications). There are $R$ separate sets of probabilities (each of which must sum to one) associated with each contingency table. Each such $theta_r$ undergoes a multidimensional logistic transformation, using the last (right-most) column as the reference category. This results in $R$ transformed vectors of dimension $(C-1)$; the transformed vectors, denoted $\omega_r s$, are stacked to form a single $\omega$ vector corresponding to that contingency table. The omega vectors are assumed to follow (i.i.d.) a multivariate normal distribution.

The posterior distribution of the THETAs/OMEGAs are in non-standard form. To sample from the posterior, the algorithm uses a Metropolis-Hastings step with a multivariate $t_4$ proposal distribution. The covariance matrix of this multivariate $t_4$ must be expanded or shrunk to achieve acceptance ratios of between .2 and .5. Tune implements `num.runs` sets of `num.iters` iterations of the Gibbs sampler. At the end of each set of iterations, Tune examines the acceptance ratios in each precinct and adjusts a shrinkage factor (a scalar multiplied to the covariance matrix of the $t_4$ proposal) upwards or downwards. When finished, Tune returns a vector of length `I` = the number of contingency tables in `data`, This vector, called `rhos`, should be fed into the `Analyze` function. See Examples here and accompanying `Analze`.

## Value

A list with the following elements.

`rhos` A vector of length `I` = number of contingency tables: each element of the `rhos` vector is a multiplier used in the proposal distribution of for draws from the conditional posterior of the THETAs, as described above. Feed this vector into the `Analyze` function.

`acc.t` Matrix of dimension `I x num.runs`: Each column of `acc.t` contains the acceptance fractions for the Metropolis-Hastings algorithm, with a multivariate $t_4$ proposal distribution, used to draw from the conditional posterior of the `THETA`s. If `Tune` has worked properly, all elements of the final column of this matrix should be between .2 and .5.

`acc.Diri` Matrix of dimension `I x num.runs`: Each column of `acc.t` contains the acceptance fractions for the Metropolis-Hastings algorithm, with independent Dirichlet proposals, used to draw from the conditional posterior of the `THETA`s. `Tune` does not alter this algorithm.

`vld.NNs` A list of length `num.runs`: Each element of `vld.NNs` is a matrix of dimension `I` by `R`, with each element of the list corresponding to one of the `num.iters` sets of iterations run by `Tune`. To draw from the conditional posterior of the internal cell counts of a contigency table, the `Tune` function draws R-1 vectors of lenth C from multinomial distributions. In then calculates the counts in the additional row (denote this row as r') deterministically. This procedure can result in negative values in row r', in which case the overall proposal for the interior cell counts is outside the parameter space (and thus invalid). Each matrix of vld.NNs keeps track of the percentage of proposals drawn in this manner that are valid (*i.e.*, not invalid). Each row of such a matrix corresponds to a contingency table. Each column in the matrix corresponds to a *row* in the a contingency table. Each entry specifies the percentage of multinomial proposals that are valid when the specified contingency table row serves as the r' row. For instance, in position 5,2 of vld.NNs is the fraction of valid proposals for the 5th contingency table when the second contigency table row is the r'th row. A value of "NaN" means that `Tune` chose to use a different (slower) method of drawing the internal cell counts because it suspected that the multinomial method would behave badly.

`acc.NNs` A list of length `num.runs`: Same as vld.NNs, except the entries represent the fraction of proposals accepted (instead of the fraction that are in the permissible parameter space).

### Author(s)

D. James Greiner, Paul D. Baines, \& Kevin M. Quinn

### References

D. James Greiner \& Kevin M. Quinn. 2009. "R x C Ecological Inference: Bounds, Correlations, Flexibility, and Transparency of Assumptions." *J.R. Statist. Soc. A* 172:67-81.

### Examples

```
## Not run:
library(RxCEcolInf)
data(stlouis)
Tune.stlouis <- Tune("Bosley, Roberts, Ribaudo, Villa, NoVote ~ bvap, ovap",
                     data = stlouis,
                     num.iters = 10000,
                     num.runs = 15)

## End(Not run)
```

---

## TuneWithExitPoll *Tuning Function for Ecological Inference for Sets of R x C Contingency Tables When Incorporating a Survey such as an Exit Poll*

---

### Description

This function tunes the markov chain monte carlo algorithm used to fit a hierarchical model to data from two sources: (i) ecological data in which the underlying contigency tables can have any number of rows or columns, and (ii data from a survey samlpe of some of the contingency tables. The user supplies the data and may specify hyperprior values. The function's primary output is a vector of multipliers, called rhos, used to adjust the covariance matrix of the multivariate $t_4$ distribution used to propose new values of intermediate-level parameters (denoted THETAS).

### Usage

```
TuneWithExitPoll(fstring, exitpoll, data = NULL, num.runs = 12,
                 num.iters = 10000, rho.vec = rep(0.05, ntables),
                 kappa = 10, nu = (mu.dim + 6), psi = mu.dim,
                 mu.vec.0 = rep(log((0.45/(mu.dim - 1))/0.55), mu.dim),
                 mu.vec.cu = runif(mu.dim, -3, 0), nolocalmode = 50,
                 sr.probs = NULL, sr.reps = NULL, numscans = 1,
                 Diri = 100, dof = 4, debug = 1)
```

**Arguments**

`fstring` String: model formula of contingency tables' column totals versus row totals. Must be in specified format (an R character string and NOT a true R formula).

`exitpoll` Matrix of dimensions $I$ = number of contingency tables = number of rows in `data` by (R * C) = number of cells in each contingency table: The results of a survey sample of some of the contingency tables. Must be in specified format. See Details.

`data` Data frame.

`num.runs` Positive integer: The number of runs or times (each of `num.iters` iterations) the tuning algorthm will be implemented.

`num.iters` Positive integer: The number of iterations in each run of the tuning algorithm.

`rho.vec` Vector of dimension $I$ = number of contigency tables = number of rows in `data`: initial values of multipliers (usually in (0,1)) to the covariance matrix of the proposal distribution for the draws of the intermediate level parameters. The purpose of this `Tune` function is to adjust these values so as to achieve acceptance ratios of between .2 and .5 in the MCMC draws of the `THETA`s.

`kappa` Scalar: The diagonal of the covariance matrix for the (normal) hyperprior distribution for the $\mu$ parameter.

`nu` Scalar: The degrees of freedom for the (Inverse-Wishart) hyperprior distriution for the `SIGMA` parameter.

`psi` Scalar: The diagonal of the matrix parameter of the (Inverse-Wishart) hyperprior distribution for the `SIGMA` parameter.

`mu.vec.0` Vector: mean of the (normal) hyperprior distribution for the $\mu$ parameter.

`mu.vec.cu` Vector of dimension $R * (C - 1)$, where $R(C)$ is the number of rows(columns) in each contigency table: Optional starting values for $\mu$ parameter.

`nolocalmode` Positive integer: How often an alternative drawing method for the contigency table internal cell counts will be used. Use of default value recommended.

`sr.probs` Matrix of dimension $I$ x $R$: Each value represents the probability of selecting a particular contingency table's row as the row to be calculated deterministically in (product multinomial) proposals for Metropolis draws of the internal cell counts. For example, if R = 3 and row 2 of position `sr.probs` = c(.1, .5, .4), then in the third contingency table (correspoding to the third row of `data`), the proposal algorithm for the interior cell counts will calculate the third contingency table's first row deterministically with probability .1, the second row with probability .5, and the third row with probability .4. Use of default (generated internally) recommended.

`sr.reps` Matrix of dimension $I$ x $R$: Each value represents the number of times the (product multinomial proposal) Metropolis algorithm will be attempted when, in drawing the internal cell counts, the proposal for the corresponding contingency table row is to be calculated deterministically. sr.reps has the same structure as sr.probs, *i.e.*, position [3,1] of sr.reps corresponds to the third contingency table's first row. Use of default (generated internally) recommended.

`numscans` Positive integer: How often the algorithm to draw the contingency table internal cell counts will be implemented before new values of the other parameters are drawn. Use of default value recommended.

`Diri` Positive integer: How often a product Dirichlet proposal distribution will be used to draw the contingency table row probability vectors (the THETAS).

`dof` Positive integer: The degrees of freedom of the multivariate $t$ proposal distribution used in drawing the contingency table row probability vectors (the THETAS).

`debug` Integer: Akin to `verbose` in some packages. If set to 1, certain status information (including rough notification regarding the number of iterations completed) will be written to the screen.

## Details

`TuneWithExitPoll` is a necessary precursor function to `AnalyzeWithExitPoll`, the workhorse function in fitting the R x C ecological inference model described in Greiner & Quinn (2009) to ecological data augmented by data from surveys of some of the contingency tables. Details and terminology of the basic (*i.e.*, without a survey sample) data structure and ecological inference model are discussed in the documentation accompanying the `Analyze` function. The purpose of `TuneWithExitPoll`, as prepatory to `AnalyzeWithExitPoll`, is the same as the purpose of `Tune` as prepatory to `Analyze`. See the documentation for `Tune` for a full explanation.

In the present implementation, the `AnalyzeWithExitPoll`, and thus `TuneWithExitPoll`, presume that the survey consisted of a simple random sample from the in-sample contingency tables. Future implementations will allow incorporation of more complicated survey sampling schemes.

The arguments to `TuneWithExitPoll` are essentially identical to those of `Tune` with the major exception of `exitpoll`. `exitpoll` feeds the results of the survey sample to the function, and a particular format is required. Specifically, `exitpoll` must have the same number of rows as `data`, meaning one row for each contigency table in the dataset. It must have R * C columns, meaning one column for each cell in one of the ecological data's contingency tables. The first row of `exitpoll` must correspond to the first row of `data`, meaning that the two rows must contain information from the same contingency table. The second row of `exitpoll` must contain information from the contingency table represented in the second row of `data`. And so on. Finally, `exitpoll` must have counts from the sample of the contingency table in vectorized row major format.

To illustrate with a voting example: Suppose the contingency tables have two rows, labeled bla and whi, and three columns, denoted Dem, Rep, and Abs. In other words, the `fstring` argument would be `"Dem, Rep, Abs ~  bla, whi"`. Suppose there are 100 contingency tables. The `data` will be of dimension $100 \times 5$, with each row consisting of the row and column totals from that particular contigency table. `exitpoll` will be of dimension $100 \times 6$. Row $11$ of the `exitpoll` will consist of the following: in position 1, the number of blacks voting Democrat observed in the sample of contingency table 11; in position 2, the number of blacks voting Republican observed in the sample of contigency table 11; in position 3, the number of blacks Abstaining from voting observed in the sample of contingency table 11; in position 4, the number of whites voting Democrat observed in the sample of contingency table 11; etc.

For tables in which there was no sample taken (*i.e.*, out-of-sample tables), the corresponding row of `exitpoll` should have a vector of 0s.

## Value

A list with the following elements.

`rhos` A vector of length `I` = number of contingency tables: each element of the `rhos` vector is a multiplier used in the proposal distribution of for draws from the conditional posterior of the THETAs, as described above. Feed this vector into the `Analyze` function.

`acc.t` Matrix of dimension `I` x `num.runs`: Each column of `acc.t` contains the acceptance fractions for the Metropolis-Hastings algorithm, with a multivariate $t_4$ proposal distribution, used to draw from the conditional posterior of the `THETAs`. If `Tune` has worked properly, all elements of the final column of this matrix should be between .2 and .5.

`acc.Diri` Matrix of dimension `I` x `num.runs`: Each column of `acc.t` contains the acceptance fractions for the Metropolis-Hastings algorithm, with independent Dirichlet proposals, used to draw from the conditional posterior of the `THETAs`. `Tune` does not alter this algorithm.

`vld.NNs` A list of length `num.runs`: Each element of `vld.NNs` is a matrix of dimension `I` by `R`, with each element of the list corresponding to one of the `num.iters` sets of iterations run by `Tune`. To draw from the conditional posterior of the internal cell counts of a contigency table, the `Tune` function draws R-1 vectors of lenth C from multinomial distributions. In then calculates the counts in the additional row (denote this row as r') deterministically. This procedure can result in negative values in row r', in which case the overall proposal for the interior cell counts is outside the parameter space (and thus invalid). Each matrix of vld.NNs keeps track of the percentage of proposals drawn in this manner that are valid (*i.e.*, not invalid). Each row of such a matrix corresponds to a contingency table. Each column in the matrix corresponds to a *row* in the a contingency table. Each entry specifies the percentage of multinomial proposals that are valid when the specified contingency table row serves as the r' row. For instance, in position 5,2 of vld.NNs is the fraction of valid proposals for the 5th contingency table when the second contigency table row is the r'th row. A value of "NaN" means that `Tune` chose to use a different (slower) method of drawing the internal cell counts because it suspected that the multinomial method would behave badly.

`acc.NNs` A list of length `num.runs`: Same as vld.NNs, except the entries represent the fraction of proposals accepted (instead of the fraction that are in the permissible parameter space).

## Author(s)

D. James Greiner, Paul D. Baines, \& Kevin M. Quinn

## References

D. James Greiner \& Kevin M. Quinn. 2009. "R x C Ecological Inference: Bounds, Correlations, Flexibility, and Transparency of Assumptions." *J.R. Statist. Soc. A* 172:67-81.

## Examples

```
## Not run:
SimData <- gendata.ep()    #  simulated data
FormulaString <- "Dem, Rep, Abs ~ bla, whi, his"
```

```
EPInvTune <-  TuneWithExitPoll(fstring = FormulaString,
                               data = SimData$GQdata,
                               exitpoll=SimData$EPInv$returnmat.ep,
                               num.iters = 10000,
                               num.runs = 15)

## End(Not run)
```

# Index

∗Topic **datasets**
    stlouis, 20
    texas, 21
∗Topic **models**
    Analyze, 3
    AnalyzeWithExitPoll, 10
    RxCEcolInf-package, 2
    Tune, 22
    TuneWithExitPoll, 25

Analyze, 3
AnalyzeWithExitPoll, 10

gendata.ep, 15

RxCEcolInf *(*RxCEcolInf-package*)*, 2
RxCEcolInf-package, 2

stlouis, 20

texas, 21
Tune, 22
TuneWithExitPoll, 25