# Package 'multiMarker'

December 11, 2020

**Version** 1.0.1

**Date** 2020-12-10

**Title** Latent Variable Model to Infer Food Intake from Multiple Biomarkers

**Description**
A latent variable model based on factor analytic and mixture of experts models, designed to infer food intake from multiple biomarkers data. The model is framed within a Bayesian hierarchical framework, which provides flexibility to adapt to different biomarker distributions and facilitates inference on food intake from biomarker data alone, along with the associated uncertainty. Details are in D'Angelo, et al. (2020) <arXiv:2006.02995>.

**Maintainer** Silvia D'Angelo <silvia.dangelo@ucd.ie>

**Imports** truncnorm, ordinalNet

**Depends** R (>= 3.0)

**License** GPL (>= 2)

**Encoding** UTF-8

**ByteCompile** true

**LazyData** true

**NeedsCompilation** no

**Author** Silvia D'Angelo [aut, cre],
Claire Gormley [ctb],
Lorraine Brennan [ctb]

**Repository** CRAN

**Date/Publication** 2020-12-11 11:00:03 UTC

## R topics documented:

multiMarker *A latent variable model to infer food intake from multiple biomarkers.*

## Description

Implements the multiMarker model via an MCMC algorithm.

## Usage

```
multiMarker(y, quantities,
            niter = 10000, burnIn = 3000,
            posteriors = FALSE, sigmaAlpha = 1,
            nuZ1 = NULL, nuZ2 = NULL,
            nuSigmaP1 = NULL, nuSigmaP2 = NULL, sigmaWprior = 0.000001,
            nuBeta1 = 2, nuBeta2 = 3, tauBeta = 0.1)
```

## Arguments

| | |
|---|---|
| y | A matrix of dimension $(n \times P)$ storing $P$ biomarker measurements on a set of $n$ observations. Missing values (NA) are allowed. |
| quantities | A vector of length $n$ storing the food quantities allocated to each of the $n$ observations in the intervention study data. Missing values (NA) are not allowed. |
| niter | The number of MCMC iterations. The default value is niter = 10000. |
| burnIn | A numerical value, the number of iterations of the chain to be discarded when computing the posterior estimates. The default value is burnIn = 3000. |
| posteriors | A logical value indicating if the full parameter chains should also be returned in output. The default value is posteriors = FALSE. |
| sigmaAlpha | Intercepts' hyperparameter ($\sigma_{\alpha^2}$), see details. The default value is sigmaAlpha = 1. |
| nuZ1, nuZ2 | Are two vectors of length $D$ storing hyperparameters for the components' variance parameters. The default values are nuZ1 = nuZ2 = NULL, corresponding to nuZ1 = (D,D-1,...,1) and nuZ2 = (D,D,...,D) . |
| nuSigmaP1, nuSigmaP2 | Scalar hyperparameters for the error's variance parameters. The default values are nuSigmaP1 = nuSigmaP2 = NULL, corresponding to nuSigmaP1 = 1 and nuSigmaP2 = n. |
| sigmaWprior | A scalar corresponding to the components' weights hyperparameter. The default value is sigmaWprior = 0.1 |
| nuBeta1, nuBeta2 | Scalar hyperparameters for the scaling coefficient's variance parameters. The default values are nuBeta1 = 2 and nuBeta2 = 3. |
| tauBeta | A scalar factor for the scaling coefficient's variance parameters. The default value is tauBeta = 0.1. |

## Details

The function facilitates inference of food intake from multiple biomarkers via MCMC, according to the multiMarker model (D'Angelo et al., 2020). The multiMarker model first learns the relationship between the multiple biomarkers and food quantity data from an intervention study and subsequently allows inference on the latent intake when only biomarker data are available.

Consider a biomarker matrix $\mathbf{Y}$ of dimension $(n \times P)$, storing $P$ different biomarker measurements on $n$ independent observations. The number of food quantities considered in the intervention study is denoted by $D$, with the corresponding set being $\mathbf{X} = (X_1, \ldots, X_d, \ldots, X_D)$ and $X_d < X_{d+1}$.

We assume that the biomarker measurements are related to an unobserved, continuous intake value, leading to the following factor analytic model:

$$y_{ip} = \alpha_p + \beta_p z_i + \epsilon_{ip}, \quad \forall \quad i = 1, \ldots, n, \quad p = 1, \ldots, P,$$

where the latent variable $z_i$ denotes the latent intake of observation $i$, with $\mathbf{z} = (z_1, \ldots, z_i, \ldots, z_n)$. The $\alpha_p$ and $\beta_p$ parameters characterize, respectively, the intercept and the scaling effect for biomarker $p$. We assume that these parameters are distributed a priori according to 0-truncated Gaussian distributions, with parameters $(\mu_\alpha, \sigma^2_\alpha)$ and $(\mu_\beta, \sigma^2_\beta)$ respectively. The error term $\epsilon_p$ is the variability associated with biomarker $p$. We assume that these errors are normally distributed with 0 mean and variance $\sigma^2_p$, which serves as a proxy for the precision of the biomarker.

A mixture of $D$ 0-truncated Gaussian distributions is assumed as prior distribution for the latent intakes. Components are centered around food quantity values $X_d$, and component-specific variances $\theta^2_d$ model food quantity-specific intake variability, with lower values suggesting higher consumption-compliance. Mixture weights are observation-specific and denoted with $\pi_i = (\pi_{i1}, \ldots, \pi_{iD})$. Given the inherent ordering of the food quantities in the intervention study, an ordinal regression model with Cauchit link function is employed to model the observation-specific weights.

A Bayesian hierarchical framework is employed for the modelling process, allowing quantification of the uncertainty in intake estimation, and flexibility in adapting to different biomarker data distributions. The framework is implemented through a Metropolis within Gibbs Markov chain Monte Carlo (MCMC) algorithm. Hyperprior distributions are assumed on the prior parameters with the corresponding hyperparameter values fixed based on the data at hand, following an empirical Bayes approach.

For more details on the estimation of the multiMarker model, see D'Angelo et al. (2020).

## Value

An object of class `'multiMarker'` containing the following components:

`estimates` A list with 9 components, storing posterior estimates of medians, standard deviations and $95\%$ credible interval lower and upper bounds for the model parameters:

- `ALPHA_E` is a matrix of dimension $(4 \times P)$ storing the posterior estimates of medians (1st row), standard deviations (2nd row) and $95\%$ credible interval lower (3rd row) and upper bounds (4th row) for the $P$ intercept parameters, $(\alpha_1, \ldots, \alpha_P)$.
- `BETA_E` is a matrix of dimension $(4 \times P)$ storing the posterior estimates of medians (1st row), standard deviations (2nd row) and $95\%$ credible interval

lower (3rd row) and upper bounds (4th row) for the $P$ scaling coefficient parameters, $(\beta_1, \ldots, \beta_P)$.

- `SigmaErr_E` is a matrix of dimension $(4 \times P)$ storing the posterior estimates of medians (1st row), standard deviations (2nd row) and $95\%$ credible interval lower (3rd row) and upper bounds (4th row) for the $P$ error variance parameters, $(\sigma_1^2, \ldots, \sigma_P^2)$.
- `SigmaD_E` is a matrix of dimension $(4 \times D)$ storing the posterior estimates of medians (1st row), standard deviations (2nd row) and $95\%$ credible interval lower (3rd row) and upper bounds (4th row) for the $D$ components' variance parameter, $(\sigma_1^2, \ldots, \sigma_D^2)$.
- `Z_E` is a matrix of dimension $(4 \times n)$ storing the posterior estimates of medians (1st row), standard deviations (2nd row) and $95\%$ credible interval lower (3rd row) and upper bounds (4th row) for the $n$ latent intakes, $(z_1, \ldots, z_n)$.
- `THETA_Est` is an array of $((P+1) \times (D-1) \times 4)$ dimensions composed of $4$ $((P+1) \times (D-1))$ matrices, storing the posterior estimates of medians (1st matrix), standard deviations (2nd matrix) and $95\%$ credible interval lower (3rd matrix) and upper bounds (4th matrix) for the components' weights parameters. In each matrix, the first row reports the values for the components' weights intercept parameter, while the other $P$ rows store those of the weights scaling coefficient parameters, $(\gamma, \theta_1, \ldots, \theta_{D-1})$.
- `sigmaBeta_E` is a vector containing the posterior estimates of medians, standard deviations and $95\%$ credible interval lower and upper bounds for the scaling coefficients' variance parameter $(\sigma_\beta^2)$.
- `muAlpha_E` is a vector containing the posterior estimates of medians, standard deviations and $95\%$ credible interval lower and upper bounds for the intercepts' mean parameter $(\mu_\alpha)$.
- `muBeta_E` is a vector containing the posterior estimates of medians, standard deviations and $95\%$ credible interval lower and upper bounds for the scaling coefficients' mean parameter $(\mu_\beta)$.
- `varPHp` Estimated error variance parameter values, $(\nu_{P1}^*, \nu_{P2}^*)$, see References.

`constants` A list with 11 components, storing constant model quantities:

- `nuZ1,nuZ2` are two vectors of length $D$ storing hyperparameters for the components' variance parameters, see References.
- `sigmaAlpha` is a scalar and it corresponds to the variance of the intercept parameters $(\sigma_{\alpha^2})$.
- `nuSigmaP1,nuSigmaP2` are scalar hyperparameters for the error's variance parameters, see References.
- `nuBeta1,nuBeta2` are scalar hyperparameters for the scaling coefficient's variance parameters, see References.
- `tauBeta` is a scalar factor for the scaling coefficient's variance parameters, see References.
- `x_D` is a vector storing the values for the $D$ food quantities.
- `P` is a scalar indicating the number of biomarkers in the data.
- `D` is a scalar indicating the number of food quantities in the data.

- `n` is a scalar indicating the number of observations the data.
- `sigmaWprior` is a scalar and it corresponds to the components' weights hyperparameter, see References.
- `y_Median` is a vector storing the observed $P$ biomarker median values.
- `y_Var` is a vector storing the observed $P$ biomarker variance values.

`chains` If `posteriors = TRUE`, a list with posterior distributions of model parameters is returned:

- `ALPHA_c` is a matrix of dimension $(niter - burnIn) \times P$ containing the estimated posterior distributions for the intercept parameters, $(\alpha_1, \ldots, \alpha_P)$.
- `BETA_c` is a matrix of dimension $(niter - burnIn) \times P$ containing the estimated posterior distributions for the scaling coefficient parameters, $(\beta_1, \ldots, \beta_P)$.
- `SigmaErr_c` is a matrix of dimension $(niter - burnIn) \times P$ containing the estimated posterior distributions for the error variance parameters, $(\sigma_1^2, \ldots, \sigma_P^2)$.
- `SigmaD_c` is a matrix of dimension $(niter - burnIn) \times D$ containing are the estimated posterior distributions for the components' variance parameters, $(\sigma_1^2, \ldots, \sigma_D^2)$.
- `Z_c` is a matrix of dimension $(niter - burnIn) \times n$ containing the estimated posterior distributions for the latent intakes, $(z_1, \ldots, z_n)$.
- `THETA_c` is an array of $(P + 1) \times (D - 1) \times (niter - burnIn)$ dimensions containing the estimated posterior distributions for the components' weights parameters. The first one corresponds to that of the weights intercept parameter, while the other $P$ posterior distributions are those of the weights scaling coefficient parameters. In each one of the $(niter - burnIn)$ matrices, the first row reports the values for the components' weights intercept parameter, while the other $P$ rows store those of the weights scaling coefficient parameters, $(\gamma, \theta_1, \ldots, \theta_{D-1})$.
- `sigmaBeta_c` is a vector containing the estimated posterior distribution for the scaling coefficients' variance parameter ($\sigma_\beta^2$).
- `muAlpha_c` is a vector containing the estimated posterior distribution for the intercepts' mean parameter ($\mu_\alpha$).
- `muBeta_c` is a vector containing the estimated posterior distribution for the scaling coefficients' mean parameter ($\mu_\beta$).
- `weights_info` is a list containing the acceptance probability values for the weights' parameters, $(\gamma, \theta_1, \ldots, \theta_{D-1})$.

## References

D'Angelo, S. and Brennan, L. and Gormley, I.C. (2020). Inferring food intake from multiple biomarkers using a latent variable model. *arxiv*

## Examples

```
library(truncnorm)
oldpar <- par(no.readonly =TRUE)
```

```
#-- Simulate intervention study biomarker and food quantity data --#

P <- D <- 3; n <- 50
alpha <- rtruncnorm(P, 0, Inf, 4, 1)
beta <- rtruncnorm(P, 0, Inf, 0.001, 0.1)
x <- c(50, 100, 150)
labels_z <- sample(c(1,2,3), n, replace = TRUE)
quantities <- x[labels_z]
sigma_d <- 8
z <- rtruncnorm(n, 0, Inf, x[labels_z], sigma_d)
Y <- sapply( 1:P, function(p) sapply( 1:n, function(i)
  max(0, alpha[p] + beta[p]*z[i] + rnorm( 1, 0, 5) ) ) )

#-- Visualize the data --#
par(mfrow= c(2,2))
boxplot(Y[,1] ~ quantities, xlab = "Food quantity", ylab = "Biomarker 1")
boxplot(Y[,2] ~ quantities, xlab = "Food quantity", ylab = "Biomarker 2")
boxplot(Y[,3] ~ quantities, xlab = "Food quantity", ylab = "Biomarker 3")

#-- Fit the multiMarker model --#
# Number of iterations (and burnIn) set small for example.
modM <- multiMarker(y = Y, quantities = quantities,
                    niter = 100, burnIn = 30,
                    posteriors = TRUE)
                    # niter and burnIn values are low only for example purposes

#-- Extract summary statistics for model parameters --#
modM$estimates$ALPHA_E[,3] #estimated median, standard deviation,
# 0.025 and 0.975 quantiles for the third intercept parameter (alpha_3)

modM$estimates$BETA_E[,2] #estimated median, standard deviation,
# 0.025 and 0.975 quantiles for the second scaling parameter (beta_2)

#-- Examine behaviour of MCMC chains --#
par(mfrow= c(2,1))
plot(modM$chains$ALPHA_c[,3], type = "l",
xlab = "Iteration (after burnin)", ylab = expression(alpha[3]) )
abline( h = mean(modM$chains$ALPHA_c[,3]), lwd = 2, col = "darkred")

plot(modM$chains$BETA_c[,2], type = "l",
xlab = "Iteration (after burnin)", ylab = expression(beta[2]) )
abline( h = mean(modM$chains$BETA_c[,2]), lwd = 2, col = "darkred")

# compute Effective Sample Size
# library(LaplacesDemon)
# ESS(modM$chains$ALPHA_c[,3]) # effective sample size for alpha_3 MCMC chain
# ESS(modM$chains$BETA_c[,2]) # effective sample size for beta_2 MCMC chain

par(oldpar)
```

`predict.multiMarker` *A latent variable model to infer food intake from multiple biomarker data alone.*

---

## Description

Implements the multiMarker model via an MCMC algorithm.

## Usage

```
## S3 method for class 'multiMarker'
predict( object, y,
         niter = 10000, burnIn = 3000,
         posteriors = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `object` | An object of class inheriting from `'multiMarker'`. |
| `y` | A matrix of dimension $(n^* \times P)$ storing $P$ biomarker measurements on a set of $n^*$ observations. Missing values (`NA`) are allowed. |
| `niter` | The number of MCMC iterations. The default value is `niter = 10000`. |
| `burnIn` | A numerical value, the number of iterations of the chain to be discarded when computing the posterior estimates. The default value is `burnIn = 3000`. |
| `posteriors` | A logical value indicating if the full parameter chains should also be returned in output. The default value is `posteriors = FALSE`. |
| `...` | Further arguments passed to or from other methods. |

## Details

The function facilitates inference on food intake from multiple biomarkers alone via MCMC, according to the multiMarker model (D'Angelo et al., 2020).

A Bayesian framework is employed for the modelling process, allowing quantification of the uncertainty associated with inferred intake. The framework is implemented through an MCMC algorithm.

For more details, see D'Angelo et al. (2020).

## Value

A list with 2 components:

`inferred_E` a list with 2 components, storing estimates of medians, standard deviations and $95\%$ credible interval lower and upper bounds for:

- `inferred_intakes` is a matrix of dimension $(4 \times n^*)$, storing the estimates of medians (1st row), standard deviations (2nd row) and $95\%$ credible interval lower (3rd row) and upper bounds (4th row) from the conditional distribution of the $n^*$ latent intakes, $(z_1^*, \ldots, z_{n^*})$.

- `inferred_Prob` is an array of dimension $(n^* \times D \times 4)$, storing estimated median (1st matrix), standard deviation (2nd matrix) and 95% credible interval lower (3rd matrix) and upper bound (4th matrix) values for the food quantity probabilities, for each one of the new $n^*$ observations.

`chains` If `posteriors = TRUE`, it contains a list with conditional distributions for:

- `ZINF` is a matrix of dimension $n^* \times niter$ containing samples from the conditional distributions of the latent intakes, $(z_1^*, \ldots, z_{n^*})$.
- `PROBS` is an array of $n^* \times D \times niter$ dimensions containing samples from the conditional distribution for food quantity probabilities, for each observation and food quantity.

## References

D'Angelo, S. and Brennan, L. and Gormley, I.C. (2020). Inferring food intake from multiple biomarkers using a latent variable model. arXiv.

## Examples

```
library(truncnorm)
oldpar <- par(no.readonly =TRUE)

#-- Simulate intervention study biomarker and food quantity data --#

P <- D <- 3; n <- 50
alpha <- rtruncnorm(P, 0, Inf, 4, 1)
beta <- rtruncnorm(P, 0, Inf, 0.001, 0.1)
x <- c(50, 100, 150)
labels_z <- sample(c(1,2,3), n, replace = TRUE)
quantities <- x[labels_z]
sigma_d <- 8
z <- rtruncnorm(n, 0, Inf, x[labels_z], sigma_d)
Y <- sapply( 1:P, function(p) sapply( 1:n, function(i)
  max(0, alpha[p] + beta[p]*z[i] + rnorm( 1, 0, 5) ) ) )

#-- Simulate Biomarker data only --#
nNew <- 20
labels_zNew <- sample(c(1,2,3), nNew, replace = TRUE)
zNew <- rtruncnorm(nNew, 0, Inf, x[labels_zNew], sigma_d)
YNew <- sapply( 1:P, function(p) sapply( 1:nNew, function(i)
  max(0, alpha[p] + beta[p]*zNew[i] + rnorm( 1, 0, 5) ) ) )

#-- Fit the multiMarker model to the intervention study data --#
# Number of iterations (and burnIn) set small for example.
modM <- multiMarker(y = Y, quantities = quantities,
                    niter = 100, burnIn = 30,
                    posteriors = TRUE)
                    # niter and burnIn values are low only for example purposes

#-- Extract summary statistics for model parameters --#
modM$estimates$ALPHA_E[,3] #estimated median, standard deviation,
```

```
# 0.025 and 0.975 quantiles for the third intercept parameter (alpha_3)

modM$estimates$BETA_E[,2] #estimated median, standard deviation,
# 0.025 and 0.975 quantiles for the second scaling parameter (beta_2)

#-- Examine behaviour of MCMC chains --#
par(mfrow= c(2,1))
plot(modM$chains$ALPHA_c[,3], type = "l",
xlab = "Iteration (after burnin)", ylab = expression(alpha[3]) )
abline( h = mean(modM$chains$ALPHA_c[,3]), lwd = 2, col = "darkred")

plot(modM$chains$BETA_c[,2], type = "l",
xlab = "Iteration (after burnin)", ylab = expression(beta[2]) )
abline( h = mean(modM$chains$BETA_c[,2]), lwd = 2, col = "darkred")

# compute Effective Sample Size
# library(LaplacesDemon)
# ESS(modM$chains$ALPHA_c[,3]) # effective sample size for alpha_3 MCMC chain
# ESS(modM$chains$BETA_c[,2]) # effective sample size for beta_2 MCMC chain

#-- Infer intakes from biomarker only data --#
# Number of iterations (and burnIn) set small for example.
infM <- predict(modM, y = YNew, niter = 100, burnIn = 30,
                posteriors = TRUE)
# niter and burnIn values are low only for example purpose

#-- Extract summary statistics for a given intake --#
obs_j <- 2 # choose which observation to look at
infM$inferred_E$inferred_intakes[, obs_j] #inferred median, standard deviation,
# 0.025 and 0.975 quantiles for the intake of observation obs_j

#-- Example of plot --#
par(mfrow = c(1,1))
hist(infM$chains$ZINF[obs_j, ], breaks = 50,
    ylab = "Density", xlab = "Intake",
    main = "Intake's conditional distribution",
    cex.main = 0.7,
    freq = FALSE) # Inferred condtional distribution of intake for observation obs_j
abline( v = infM$inferred_E$inferred_intakes[1,obs_j], col = "darkred",
lwd = 2 ) # median value
abline( v = infM$inferred_E$inferred_intakes[3,obs_j], col = "grey",
lwd = 2 )
abline( v = infM$inferred_E$inferred_intakes[4,obs_j], col = "grey",
lwd = 2 )
legend( x = "topleft", fill = c("grey", "darkred"), title = "quantiles:",
legend = c("(0.025, 0.975)", "0.5"), bty = "n", cex = 0.7)

mtext(paste("Observation", obs_j, sep = " "), outer = TRUE, cex = 1.5)
par(oldpar)
```

# Index