

# Package ‘psychtm’

November 2, 2021

**Type** Package

**Title** Text Mining Methods for Psychological Research

**Version** 2021.1.0

**Description** Provides text mining methods for social science research. The package implements estimation, inference, summarization, and goodness-of-fit methods for topic models including Latent Dirichlet Allocation (LDA), supervised LDA, and supervised LDA with covariates using Bayesian Markov Chain Monte Carlo. A description of the key models and estimation methods is available in Wilcox, Jacobucci, Zhang, & Ammerman (2021). <doi:10.31234/osf.io/62tc3>.

**License** LGPL (>= 3)

**URL** <https://github.com/ktw5691/psychtm/>

**BugReports** <https://github.com/ktw5691/psychtm/issues>

**Depends** R (>= 3.3.0)

**Imports** coda (>= 0.4), label.switching, methods, Rcpp (>= 0.11.0), rlang (>= 0.4.10), tibble (>= 2.1.3)

**Suggests** spelling, knitr (>= 1.22), covr, dplyr, ggplot2, lda, testthat (>= 3.0.2), rmarkdown

**LinkingTo** Rcpp (>= 0.11.0), RcppArmadillo, RcppProgress (>= 0.4.2)

**SystemRequirements** C++11

**VignetteBuilder** knitr

**Encoding** UTF-8

**Language** en-US

**LazyData** true

**RoxygenNote** 7.1.2

**Collate** 'RcppExports.R' 'aaa-classes.R' 'aaa-generics.R' 'data.R' 'deprec-functions.R' 'helper-functions.R' 'onUnload.R' 'psychtm-package.R' 'psychtm.R' 'sldax-methods.R' 'utils.R'

**NeedsCompilation** yes

**Author** Kenneth Wilcox [aut, cre, cph]

**Maintainer** Kenneth Wilcox <kwilcox3@nd.edu>

**Repository** CRAN

**Date/Publication** 2021-11-02 09:20:05 UTC

## R topics documented:

a0	3
a0<-	4
alpha	4
alpha<-	5
b0	6
b0<-	6
beta_	7
beta_<-	7
eta	8
eta<-	8
eta_start	9
eta_start<-	9
extra	10
extra<-	10
gamma_	11
gamma_<-	12
gibbs_logistic	12
gibbs_mlr	14
gibbs_sldax	15
Logistic-class	18
loglike	19
loglike<-	19
logpost	20
logpost<-	20
lpd	21
lpd<-	21
Mlr-class	22
Model-class	23
mu0	26
mu0<-	27
nchain	27
nchain<-	28
ndocs	28
ndocs<-	29
ntopics	29
ntopics<-	30
nvocab	30
nvocab<-	31
prep_docs	32
proposal_sd	33
proposal_sd<-	33

psychtm . . . . .	34
p_eff . . . . .	34
p_eff<- . . . . .	35
se_waic . . . . .	35
se_waic<- . . . . .	36
sigma0 . . . . .	36
sigma0<- . . . . .	37
sigma2 . . . . .	37
sigma2<- . . . . .	38
Sldax-class . . . . .	38
sldax-summary . . . . .	41
teacher_rate . . . . .	45
term_score . . . . .	45
theta . . . . .	46
theta<- . . . . .	47
topics . . . . .	47
topics<- . . . . .	48
waic . . . . .	48
waic<- . . . . .	49
waic_all . . . . .	50
waic_d . . . . .	50
waic_diff . . . . .	51

<b>Index</b>	<b>52</b>
--------------	-----------

---

a0	<i>Create generic a0 function for class</i>
----	---

---

## Description

Create generic a0 function for class

## Usage

a0(x)

## Arguments

x                    An Mlr object.

## Value

Double value of shape prior parameter for residual variance.

## Examples

```
m1 <- Mlr(ndocs = 1)
a0(m1)
```

---

`a0<-` *Create generic a0<- function for class*

---

**Description**

Create generic a0<- function for class

**Usage**

```
a0(x) <- value
```

**Arguments**

`x` An Mlr object.  
`value` Numeric shape parameter for residual variance prior to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Mlr(ndocs = 1)
a0(m1) <- 1.0
```

---

`alpha` *Create generic alpha function for class*

---

**Description**

Create generic alpha function for class

**Usage**

```
alpha(x)
```

**Arguments**

`x` An Sldax object.

**Value**

Double value of parameter for symmetric Dirichlet distribution prior on the topic proportions.

**Examples**

```
m1 <- Sldax(ndocs = 1, nvocab = 2,
            topics = array(c(1, 2, 2, 1), dim = c(1, 4, 1)),
            theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
            beta = array(c(0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 1)))
alpha(m1)
```

---

alpha<- *Create generic alpha<- function for class*

---

**Description**

Create generic alpha<- function for class

**Usage**

```
alpha(x) <- value
```

**Arguments**

x	An Sldax object.
value	Numeric parameter for symmetric Dirichlet prior on topic proportions to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Sldax(ndocs = 1, nvocab = 2,
            topics = array(c(1, 2, 2, 1), dim = c(1, 4, 1)),
            theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
            beta = array(c(0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 1)))
alpha(m1) <- 1.0
```

---

b0 *Create generic b0 function for class*

---

**Description**

Create generic b0 function for class

**Usage**

```
b0(x)
```

**Arguments**

x An Mlr object.

**Value**

Double value of rate prior parameter for residual variance.

**Examples**

```
m1 <- Mlr(ndocs = 1)
b0(m1)
```

---

b0<- *Create generic b0<- function for class*

---

**Description**

Create generic b0<- function for class

**Usage**

```
b0(x) <- value
```

**Arguments**

x An Mlr object.  
value Numeric value of rate parameter for residual variance prior to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Mlr(ndocs = 1)
b0(m1) <- 1.0
```

---

beta\_ *Create generic beta\_ function for class*

---

**Description**

Create generic beta\_ function for class

**Usage**

```
beta_(x)
```

**Arguments**

x An Sldax object.

**Value**

A numeric array of topic-word probability distributions across sampler iterations.

**Examples**

```
m1 <- Sldax(ndocs = 1, nvocab = 2,
            topics = array(c(1, 2, 2, 1), dim = c(1, 4, 1)),
            theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
            beta = array(c(0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 1)))
beta_(m1)
```

---

beta\_<- *Create generic beta\_<- function for class*

---

**Description**

Create generic beta\_<- function for class

**Usage**

```
beta_(x) <- value
```

**Arguments**

x An Sldax object.  
value Numeric array of topic-word probabilities to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Sldax(ndocs = 1, nvocab = 2,
           topics = array(c(1, 2, 2, 1), dim = c(1, 4, 1)),
           theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
           beta = array(c(0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 1)))
beta_(m1) <- array(c(0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 1))
```

---

eta *Create generic eta function for class*

---

**Description**

Create generic eta function for class

**Usage**

```
eta(x)
```

**Arguments**

x An Model object.

**Value**

A numeric matrix of posterior draws of regression coefficients.

**Examples**

```
m1 <- Model(ndocs = 1)
eta(m1)
```

---

eta<- *Create generic eta<- function for class*

---

**Description**

Create generic eta<- function for class

**Usage**

```
eta(x) <- value
```

**Arguments**

x An Model object.  
value Numeric vector of regression coefficients to assign to slot.



**Value**

None.

**Examples**

```
m1 <- Model(ndocs = 1)
eta(m1) <- matrix(c(-1.0, 1.0), nrow = 1, ncol = 2)
```

---

eta_start	<i>Create generic eta_start function for class</i>
-----------	--

---

**Description**

Create generic eta\_start function for class

**Usage**

```
eta_start(x)
```

**Arguments**

x                    An Model object.

**Value**

Numeric vector of starting values for regression coefficients.

**Examples**

```
m1 <- Model(ndocs = 1)
eta_start(m1)
```

---

eta_start<-	<i>Create generic eta_start&lt;- function for class</i>
-------------	---

---

**Description**

Create generic eta\_start<- function for class

**Usage**

```
eta_start(x) <- value
```

**Arguments**

x                    An Model object.  
value                Numeric vector of starting values for regression coefficients to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Model(ndocs = 1)
eta_start(m1) <- rep(0.0, times = 2)
```

---

extra	<i>Create generic extra function for class</i>
-------	--

---

**Description**

Create generic extra function for class

**Usage**

```
extra(x)
```

**Arguments**

x                    An Model object.

**Value**

A list of model fitting information including time elapsed, label switching correction status, and the original function call.

**Examples**

```
m1 <- Model(ndocs = 1)
extra(m1)
```

---

extra<-	<i>Create generic extra&lt;- function for class</i>
---------	---

---

**Description**

Create generic extra<- function for class

**Usage**

```
extra(x) <- value
```

**Arguments**

x	An Model object.
value	List of additional model fitting information to assign to slot.

**Value**

None.

---

gamma\_ *Create generic gamma\_ function for class*

---

**Description**

Create generic gamma\_ function for class

**Usage**

```
gamma_(x)
```

**Arguments**

x	An Sldax object.
---	------------------

**Value**

Double value of parameter for symmetric Dirichlet distribution prior on the topic-word probabilities.

**Examples**

```
m1 <- Sldax(ndocs = 1, nvocab = 2,
            topics = array(c(1, 2, 2, 1), dim = c(1, 4, 1)),
            theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
            beta = array(c(0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 1)))
gamma_(m1)
```

---

gamma\_<- *Create generic gamma\_<- function for class*

---

**Description**

Create generic gamma\_<- function for class

**Usage**

```
gamma_(x) <- value
```

**Arguments**

x	An Sldax object.
value	Numeric parameter for symmetric Dirichlet prior on topic-word probabilities to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Sldax(ndocs = 1, nvocab = 2,
            topics = array(c(1, 2, 2, 1), dim = c(1, 4, 1)),
            theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
            beta = array(c(0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 1)))
gamma_(m1) <- 1.0
```

---

gibbs\_logistic *Fit logistic regression model*

---

**Description**

gibbs\_logistic() is used to fit a Bayesian logistic regression model using Gibbs sampling.

**Usage**

```
gibbs_logistic(
  formula,
  data,
  m = 100,
  burn = 0,
  thin = 1,
  mu0 = NULL,
  sigma0 = NULL,
```

```

    eta_start = NULL,
    proposal_sd = NULL,
    verbose = FALSE,
    display_progress = FALSE
  )

```

## Arguments

<code>formula</code>	An object of class <a href="#">formula</a> : a symbolic description of the model to be fitted.
<code>data</code>	An optional data frame containing the variables in the model.
<code>m</code>	The number of iterations to run the Gibbs sampler (default: 100).
<code>burn</code>	The number of iterations to discard as the burn-in period (default: 0).
<code>thin</code>	The period of iterations to keep after the burn-in period (default: 1).
<code>mu0</code>	An optional $p \times 1$ mean vector for the prior on the regression coefficients. See 'Details'.
<code>sigma0</code>	A $p \times p$ variance-covariance matrix for the prior on the regression coefficients. See 'Details'.
<code>eta_start</code>	A $p \times 1$ vector of starting values for the regression coefficients.
<code>proposal_sd</code>	The proposal standard deviations for drawing the regression coefficients, $N(0, \text{proposal\_sd}(j))$ , $j = 1, \dots, p$ (default: 2.38 for all coefficients).
<code>verbose</code>	Should parameter draws be output during sampling? (default: FALSE).
<code>display_progress</code>	Show progress bar? (default: FALSE). Do not use with <code>verbose = TRUE</code> .

## Details

For `mu0`, by default, we use a vector of  $p$   $\theta$ s for  $p$  regression coefficients.

For `sigma0`, by default, we use a  $p \times p$  diagonal matrix with diagonal elements (variances) of 6.25.

## Value

An object of class [Logistic](#).

## See Also

Other Gibbs sampler: [gibbs\\_mlr\(\)](#), [gibbs\\_sldax\(\)](#)

## Examples

```

data(mtcars)
m1 <- gibbs_logistic(vs ~ hp, data = mtcars)

```

---

`gibbs_mlr`*Fit linear regression model*

---

## Description

`gibbs_mlr()` is used to fit a Bayesian linear regression model using Gibbs sampling.

## Usage

```
gibbs_mlr(  
  formula,  
  data,  
  m = 100,  
  burn = 0,  
  thin = 1,  
  mu0 = NULL,  
  sigma0 = NULL,  
  a0 = NULL,  
  b0 = NULL,  
  eta_start = NULL,  
  verbose = FALSE,  
  display_progress = FALSE  
)
```

## Arguments

<code>formula</code>	An object of class <code>formula</code> : a symbolic description of the model to be fitted.
<code>data</code>	An optional data frame containing the variables in the model.
<code>m</code>	The number of iterations to run the Gibbs sampler (default: 100).
<code>burn</code>	The number of iterations to discard as the burn-in period (default: 0).
<code>thin</code>	The period of iterations to keep after the burn-in period (default: 1).
<code>mu0</code>	An optional $p \times 1$ mean vector for the prior on the regression coefficients. See 'Details'.
<code>sigma0</code>	A $p \times p$ variance-covariance matrix for the prior on the regression coefficients. See 'Details'.
<code>a0</code>	The shape parameter for the prior on <code>sigma2</code> (default: 0.001).
<code>b0</code>	The scale parameter for the prior on <code>sigma2</code> (default: 0.001).
<code>eta_start</code>	A $p \times 1$ vector of starting values for the regression coefficients.
<code>verbose</code>	Should parameter draws be output during sampling? (default: FALSE).
<code>display_progress</code>	Show progress bar? (default: FALSE). Do not use with <code>verbose = TRUE</code> .

**Details**

For  $\mu_0$ , by default, we use a vector of  $p$  0s for  $p$  regression coefficients.

For  $\sigma_0$ , by default, we use a  $p \times p$  identity matrix.

**Value**

An object of class `Mlr`.

**See Also**

Other Gibbs sampler: `gibbs_logistic()`, `gibbs_sldax()`

**Examples**

```
data(mtcars)
m1 <- gibbs_mlr(mpg ~ hp, data = mtcars)
```

---

gibbs\_sldax

*Fit supervised or unsupervised topic models (SLDAX or LDA)*


---

**Description**

`gibbs_sldax()` is used to fit both supervised and unsupervised topic models.

**Usage**

```
gibbs_sldax(
  formula,
  data,
  m = 100,
  burn = 0,
  thin = 1,
  docs,
  V,
  K = 2L,
  model = c("lda", "slda", "sldax", "slda_logit", "sldax_logit"),
  sample_beta = TRUE,
  sample_theta = TRUE,
  interaction_xcol = -1L,
  alpha_ = 1,
  gamma_ = 1,
  mu0 = NULL,
  sigma0 = NULL,
  a0 = NULL,
  b0 = NULL,
  eta_start = NULL,
```

```

    constrain_eta = FALSE,
    proposal_sd = NULL,
    return_assignments = FALSE,
    correct_ls = TRUE,
    verbose = FALSE,
    display_progress = FALSE
  )

```

## Arguments

formula	An object of class <code>formula</code> : a symbolic description of the model to be fitted.
data	An optional data frame containing the variables in the model.
m	The number of iterations to run the Gibbs sampler (default: 100).
burn	The number of iterations to discard as the burn-in period (default: 0).
thin	The period of iterations to keep after the burn-in period (default: 1).
docs	A $D \times \max(N_d)$ matrix of word indices for all documents.
V	The number of unique terms in the vocabulary.
K	The number of topics.
model	A string denoting the type of model to fit. See 'Details'. (default: "lda").
sample_beta	A logical (default = TRUE): If TRUE, the topic-vocabulary distributions are sampled from their full conditional distribution.
sample_theta	A logical (default = TRUE): If TRUE, the topic proportions will be sampled. CAUTION: This can be memory-intensive.
interaction_xcol	EXPERIMENTAL: The column number of the design matrix for the additional predictors for which an interaction with the $K$ topics is desired (default: -1L, no interaction). Currently only supports a single continuous predictor or a two-category categorical predictor represented as a single dummy-coded column.
alpha_	The hyper-parameter for the prior on the topic proportions (default: 1.0).
gamma_	The hyper-parameter for the prior on the topic-specific vocabulary probabilities (default: 1.0).
mu0	An optional $q \times 1$ mean vector for the prior on the regression coefficients. See 'Details'.
sigma0	A $q \times q$ variance-covariance matrix for the prior on the regression coefficients. See 'Details'.
a0	The shape parameter for the prior on sigma2 (default: 0.001).
b0	The scale parameter for the prior on sigma2 (default: 0.001).
eta_start	A $q \times 1$ vector of starting values for the regression coefficients.
constrain_eta	A logical (default = FALSE): If TRUE, the regression coefficients will be constrained so that they are in descending order; if FALSE, no constraints will be applied.
proposal_sd	The proposal standard deviations for drawing the regression coefficients, $N(0, \text{proposal\_sd}(j))$ , $j = 1, \dots, q$ . Only used for <code>model = "slda_logit"</code> and <code>model = "sldax_logit"</code> (default: 2.38 for all coefficients).



return_assignments	A logical (default = FALSE): If TRUE, returns an $N \times \max N_d \times M$ array of topic assignments in slot @topics. CAUTION: this can be memory-intensive.
correct_ls	Run Stephens (2000) label switching correct algorithm on posterior? (default = TRUE).
verbose	Should parameter draws be output during sampling? (default: FALSE).
display_progress	Show progress bar? (default: FALSE). Do not use with verbose = TRUE.

## Details

The number of regression coefficients  $q$  in supervised topic models is determined as follows: For the SLDA model with only the  $K$  topics as predictors,  $q = K$ ; for the SLDAX model with  $K$  topics and  $p$  additional predictors, there are two possibilities: (1) If no interaction between an additional covariate and the  $K$  topics is desired (default: `interaction_xcol = -1L`),  $q = p + K$ ; (2) if an interaction between an additional covariate and the  $K$  topics is desired (e.g., `interaction_xcol = 1`),  $q = p + 2K - 1$ . If you supply custom values for prior parameters  $\mu_0$  or  $\sigma_0$ , be sure that the length of  $\mu_0$  ( $q$ ) and/or the number of rows and columns of  $\sigma_0$  ( $q \times q$ ) are correct. If you supply custom starting values for `eta_start`, be sure that the length of `eta_start` is correct.

For `model`, one of `c("lda", "slda", "sldax", "slda_logit", "sldax_logit")`.

- "lda": unsupervised topic model;
- "slda": supervised topic model with a continuous outcome;
- "sldax": supervised topic model with a continuous outcome and additional predictors of the outcome;
- "slda\_logit": supervised topic model with a dichotomous outcome (0/1);
- "sldax\_logit": supervised topic model with a dichotomous outcome (0/1) and additional predictors of the outcome.

For  $\mu_0$ , the first  $p$  elements correspond to coefficients for the  $p$  additional predictors (if none,  $p = 0$ ), while elements  $p + 1$  to  $p + K$  correspond to coefficients for the  $K$  topics, and elements  $p + K + 1$  to  $p + 2K - 1$  correspond to coefficients for the interaction (if any) between one additional predictor and the  $K$  topics. By default, we use a vector of  $q$   $\theta$ s.

For  $\sigma_0$ , the first  $p$  rows/columns correspond to coefficients for the  $p$  additional predictors (if none,  $p = 0$ ), while rows/columns  $p + 1$  to  $p + K$  correspond to coefficients for the  $K$  topics, and rows/columns  $p + K + 1$  to  $p + 2K - 1$  correspond to coefficients for the interaction (if any) between one additional predictor and the  $K$  topics. By default, we use an identity matrix for `model = "slda"` and `model = "sldax"` and a diagonal matrix with diagonal elements (variances) of 6.25 for `model = "slda_logit"` and `model = "sldax_logit"`.

## Value

An object of class `Sldax`.

## See Also

Other Gibbs sampler: `gibbs_logistic()`, `gibbs_mlr()`

**Examples**

```
library(lda) # Required if using `prep_docs()`

data(teacher_rate) # Synthetic student ratings of instructors
docs_vocab <- prep_docs(teacher_rate, "doc")
vocab_len <- length(docs_vocab$vocab)
m1 <- gibbs_sldax(rating ~ I(grade - 1), m = 2,
                 data = teacher_rate, docs = docs_vocab$documents,
                 V = vocab_len, K = 2, model = "sldax")
```

---

 Logistic-class

*S4 class for a logistic regression model that inherits from [Model](#)*


---

**Description**

S4 class for a logistic regression model that inherits from [Model](#)

Helper function (constructor) for Logistic class

**Usage**

```
## S4 method for signature 'Logistic'
proposal_sd(x)

## S4 replacement method for signature 'Logistic'
proposal_sd(x) <- value

Logistic(proposal_sd = NaN, ...)
```

**Arguments**

x	An Logistic object.
value	A value to assign to a slot for x
proposal_sd	A vector of p + 1 proposal scales/standard deviations for sampling of p + 1 regression coefficients by Metropolis-Hastings.
...	additional arguments to be passed to the low level regression fitting functions (see below).

**Value**

A [Logistic](#) object.

**Slots**

proposal\_sd A vector of p + 1 proposal scales/standard deviations for sampling of p + 1 regression coefficients by Metropolis-Hastings.

**Examples**

```
m1 <- Logistic(ndocs = 1)
print(m1)
```

---

**loglike***Create generic loglike function for class*

---

**Description**

Create generic loglike function for class

**Usage**

```
loglike(x)
```

**Arguments**

x                    An Model object.

**Value**

A numeric vector of log-likelihood values across sampler iterations.

**Examples**

```
m1 <- Model(ndocs = 1)
loglike(m1)
```

---

**loglike<-***Create generic loglike<- function for class*

---

**Description**

Create generic loglike<- function for class

**Usage**

```
loglike(x) <- value
```

**Arguments**

x                    An Model object.  
value                Numeric vector of log likelihoods to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Model(ndocs = 1)
loglike(m1) <- rep(NaN, times = nchain(m1))
```

---

logpost

*Create generic logpost function for class*

---

**Description**

Create generic logpost function for class

**Usage**

```
logpost(x)
```

**Arguments**

x                    An Model object.

**Value**

A numeric vector of log-posterior values across sampler iterations.

**Examples**

```
m1 <- Model(ndocs = 1)
logpost(m1)
```

---

logpost<-

*Create generic logpost<- function for class*

---

**Description**

Create generic logpost<- function for class

**Usage**

```
logpost(x) <- value
```

**Arguments**

x                    An Model object.  
value                Numeric vector of log posteriors to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Model(ndocs = 1)
logpost(m1) <- rep(NaN, times = nchain(m1))
```

---

lpd

*Create generic lpd function for class*


---

**Description**

Create generic lpd function for class

**Usage**

```
lpd(x)
```

**Arguments**

x                    An Model object.

**Value**

Numeric log-predictive density used in WAIC.

**Examples**

```
m1 <- Model(ndocs = 1)
lpd(m1)
```

---

lpd<-

*Create generic lpd<- function for class*


---

**Description**

Create generic lpd<- function for class

**Usage**

```
lpd(x) <- value
```

**Arguments**

x                    An Model object.  
value                Numeric matrix of log predictive densities in each document to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Model(ndocs = 1)
lpd(m1) <- matrix(NaN, nrow = 1, ncol = 1)
```

---

Mlr-class

*S4 class for a regression model that inherits from [Model](#).*


---

**Description**

S4 class for a regression model that inherits from [Model](#).

Helper function (constructor) for Mlr class

**Usage**

```
## S4 method for signature 'Mlr'
sigma2(x)

## S4 replacement method for signature 'Mlr'
sigma2(x) <- value

## S4 method for signature 'Mlr'
a0(x)

## S4 replacement method for signature 'Mlr'
a0(x) <- value

## S4 method for signature 'Mlr'
b0(x)

## S4 replacement method for signature 'Mlr'
b0(x) <- value

Mlr(a0 = 0.001, b0 = 0.001, sigma2 = NaN, ...)
```

**Arguments**

x	An Model object.
value	A value to assign to a slot for x
a0	A prior shape hyperparameter for sigma2.
b0	A prior rate hyperparameter for sigma2.
sigma2	A nchain x 1 numeric vector of draws of the residual variance.
...	additional arguments to be passed to the low level regression fitting functions (see below).

**Value**

An `Mlr` object.

**Slots**

`a0` A prior shape hyperparameter for `sigma2`.

`b0` A prior rate hyperparameter for `sigma2`.

`sigma2` A `nchain` x 1 numeric vector of draws of the residual variance.

**Examples**

```
m1 <- Mlr(ndocs = 1)
print(m1)
```

---

Model-class

*An S4 super class to represent a regression-like model*

---

**Description**

An S4 super class to represent a regression-like model

Helper function (constructor) for `Model` class

**Usage**

```
## S4 method for signature 'Model'
ndocs(x)
```

```
## S4 replacement method for signature 'Model'
ndocs(x) <- value
```

```
## S4 method for signature 'Model'
nchain(x)
```

```
## S4 replacement method for signature 'Model'
nchain(x) <- value
```

```
## S4 method for signature 'Model'
mu0(x)
```

```
## S4 replacement method for signature 'Model'
mu0(x) <- value
```

```
## S4 method for signature 'Model'
sigma0(x)
```

```
## S4 replacement method for signature 'Model'  
sigma0(x) <- value  
  
## S4 method for signature 'Model'  
eta_start(x)  
  
## S4 replacement method for signature 'Model'  
eta_start(x) <- value  
  
## S4 method for signature 'Model'  
eta(x)  
  
## S4 replacement method for signature 'Model'  
eta(x) <- value  
  
## S4 method for signature 'Model'  
loglike(x)  
  
## S4 replacement method for signature 'Model'  
loglike(x) <- value  
  
## S4 method for signature 'Model'  
logpost(x)  
  
## S4 replacement method for signature 'Model'  
logpost(x) <- value  
  
## S4 method for signature 'Model'  
waic(x)  
  
## S4 replacement method for signature 'Model'  
waic(x) <- value  
  
## S4 method for signature 'Model'  
se_waic(x)  
  
## S4 replacement method for signature 'Model'  
se_waic(x) <- value  
  
## S4 method for signature 'Model'  
p_eff(x)  
  
## S4 replacement method for signature 'Model'  
p_eff(x) <- value  
  
## S4 method for signature 'Model'  
lpd(x)
```



```

## S4 replacement method for signature 'Model'
lpd(x) <- value

## S4 method for signature 'Model'
extra(x)

## S4 replacement method for signature 'Model'
extra(x) <- value

Model(
  ndocs,
  nchain = 1,
  mu0 = NaN,
  sigma0 = NaN,
  eta_start = NaN,
  eta = NaN,
  loglike = NaN,
  logpost = NaN,
  waic = NaN,
  se_waic = NaN,
  p_eff = NaN,
  lpd = NaN
)

```

### Arguments

x	An Model object.
value	A value to assign to a slot for x
ndocs	The number of documents/observations.
nchain	The number of iterations of the Gibbs sampler.
mu0	A (p + 1) x 1 matrix of prior means for eta.
sigma0	A (p + 1) x (p + 1) prior covariance matrix for eta.
eta_start	A (p + 1) x 1 matrix of starting values for eta.
eta	A nchain x (p + 1) matrix of draws of regression coefficients.
loglike	A nchain x 1 vector of the log-likelihood (up to an additive constant).
logpost	A nchain x 1 vector of the log-posterior (up to an additive constant).
waic	WAIC (up to an additive constant) on the deviance scale.
se_waic	Standard error of the WAIC.
p_eff	The effective number of parameters.
lpd	A nchain x ndocs matrix of predictive posterior likelihoods.

### Value

A [Model](#) object.

**Slots**

**ndocs** The number of documents/observations.  
**nchain** The number of iterations of the Gibbs sampler.  
**mu0** A  $(p + 1) \times 1$  matrix of prior means for eta.  
**sigma0** A  $(p + 1) \times (p + 1)$  prior covariance matrix for eta.  
**eta\_start** A  $(p + 1) \times 1$  matrix of starting values for eta.  
**eta** A  $nchain \times (p + 1)$  matrix of draws of regression coefficients.  
**loglike** A  $nchain \times 1$  vector of the log-likelihood (up to an additive constant).  
**logpost** A  $nchain \times 1$  vector of the log-posterior (up to an additive constant).  
**waic** WAIC (up to an additive constant) on the deviance scale.  
**se\_waic** Standard error of the WAIC.  
**p\_eff** The effective number of parameters.  
**lpd** A  $nchain \times ndocs$  matrix of predictive posterior likelihoods.  
**extra** A list of additional model fitting information. Contains `time_elapsed`, `start_time`, `end_time`, `corrected_label_switching`, and `call`.

**Examples**

```
m1 <- Model(ndocs = 1)
print(m1)
```

---

**mu0**
*Create generic mu0 function for class*


---

**Description**

Create generic mu0 function for class

**Usage**

```
mu0(x)
```

**Arguments**

**x** An Model object.

**Value**

Numeric vector of prior means for regression coefficients eta.

**Examples**

```
m1 <- Model(ndocs = 1)
mu0(m1)
```

---

mu0<- *Create generic mu0<- function for class*

---

**Description**

Create generic mu0<- function for class

**Usage**

```
mu0(x) <- value
```

**Arguments**

x                    An Model object.  
value                Numeric vector of prior means for regression coefficients to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Model(ndocs = 1)  
mu0(m1) <- rep(0.0, times = 2)
```

---

nchain *Create generic nchain function for class*

---

**Description**

Create generic nchain function for class

**Usage**

```
nchain(x)
```

**Arguments**

x                    An Model object.

**Value**

Integer length of sampler chain.

**Examples**

```
m1 <- Model(ndocs = 1)  
nchain(m1)
```

---

nchain<- *Create generic nchain<- function for class*

---

**Description**

Create generic nchain<- function for class

**Usage**

```
nchain(x) <- value
```

**Arguments**

x	An Model object.
value	Integer length of sampler chain to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Model(ndocs = 1)
nchain(m1) <- 100
```

---

ndocs *Create generic ndocs function for class*

---

**Description**

Create generic ndocs function for class

**Usage**

```
ndocs(x)
```

**Arguments**

x	An Model object.
---	------------------

**Value**

Integer number of documents.

**Examples**

```
m1 <- Model(ndocs = 1)
ndocs(m1)
```

---

ndocs<- *Create generic ndocs<- function for class*

---

**Description**

Create generic ndocs<- function for class

**Usage**

```
ndocs(x) <- value
```

**Arguments**

x                    An Model object.  
value                Integer number of documents to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Model(ndocs = 1)  
ndocs(m1) <- 2
```

---

ntopics *Create generic ntopics function for class*

---

**Description**

Create generic ntopics function for class

**Usage**

```
ntopics(x)
```

**Arguments**

x                    An Sldax object.

**Value**

Integer number of topics in model.

**Examples**

```
m1 <- Sldax(ndocs = 1, nvocab = 2,
           topics = array(c(1, 2, 2, 1), dim = c(1, 4, 1)),
           theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
           beta = array(c(0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 1)))
ntopics(m1)
```

---

ntopics<- *Create generic ntopics<- function for class*

---

**Description**

Create generic ntopics<- function for class

**Usage**

```
ntopics(x) <- value
```

**Arguments**

x	An Sldax object.
value	Integer number of topics to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Sldax(ndocs = 1, nvocab = 2,
           topics = array(c(1, 2, 2, 1), dim = c(1, 4, 1)),
           theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
           beta = array(c(0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 1)))
ntopics(m1) <- 2
```

---

nvocab *Create generic nvocab function for class*

---

**Description**

Create generic nvocab function for class

**Usage**

```
nvocab(x)
```

**Arguments**

x                    An Sldax object.

**Value**

Integer number of unique terms in vocabulary.

**Examples**

```
m1 <- Sldax(ndocs = 1, nvocab = 2,
            topics = array(c(1, 2, 2, 1), dim = c(1, 4, 1)),
            theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
            beta = array(c(0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 1)))
nvocab(m1)
```

---

nvocab<-                    *Create generic nvocab<- function for class*

---

**Description**

Create generic nvocab<- function for class

**Usage**

```
nvocab(x) <- value
```

**Arguments**

x                    An Sldax object.  
value                Numeric number of unique terms in vocabulary to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Sldax(ndocs = 1, nvocab = 2,
            topics = array(c(1, 2, 2, 1), dim = c(1, 4, 1)),
            theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
            beta = array(c(0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 1)))
nvocab(m1) <- 2L
```

---

prep_docs	<i>Prepare documents in a data frame for modeling</i>
-----------	---

---

### Description

prep\_docs() takes documents stored as a column of a data frame and converts them into a list containing a matrix representation of documents and vocabulary character vector for modeling.

### Usage

```
prep_docs(data, col, lower = TRUE)
```

### Arguments

data	A data frame containing a column of documents.
col	A character string denoting the column of documents in data.
lower	Should all terms be converted to lowercase? (default: TRUE).

### Value

A list with two components: documents A matrix of term uses with one row per document and one column per term position up to the number of terms in the longest document; vocab A character vector of unique terms in the documents.

### Note

This function does not perform further data preprocessing such as stop-word removal. It is assumed that the unit of analysis is each term, so this function will not be appropriate for other units of analysis such as n-grams or sentences.

### Examples

```
data(teacher_rate) # Synthetic student ratings of instructors
docs_vocab <- prep_docs(teacher_rate, "doc")
str(docs_vocab) # A list with two components `documents` and `vocab`
```



---

proposal_sd	<i>Create generic proposal_sd function for class</i>
-------------	--

---

**Description**

Create generic proposal\_sd function for class

**Usage**

```
proposal_sd(x)
```

**Arguments**

x                    An Logistic object.

**Value**

Numeric vector of proposal scales for Metropolis step for regression coefficients sampling.

**Examples**

```
m1 <- Logistic(ndocs = 1)
proposal_sd(m1)
```

---

proposal_sd<-	<i>Create generic proposal_sd&lt;- function for class</i>
---------------	---

---

**Description**

Create generic proposal\_sd<- function for class

**Usage**

```
proposal_sd(x) <- value
```

**Arguments**

x                    An Logistic object.  
value                Numeric vector of scale parameters for Metropolis sampling of regression coefficients to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Logistic(ndocs = 1)
proposal_sd(m1) <- c(2.38, 2.38)
```

---

psychtm	<i>psychtm: A package for text mining methods for psychological research</i>
---------	--

---

**Description**

The psychtm package provides estimation, summarization, and goodness-of-fit functions:

**Model Fitting**

The workhorse function for Bayesian estimation of topic models is [gibbs\\_sldax\(\)](#). Similarly, see [gibbs\\_mlr\(\)](#) and [gibbs\\_logistic\(\)](#) to estimate regression models with continuous and dichotomous outcomes, respectively.

**Parameter Estimates and Goodness-of-Fit**

See [sldax-summary](#) for functions to obtain and summarize parameter estimates and to compute goodness-of-fit metrics.

---

p_eff	<i>Create generic p_eff function for class</i>
-------	--

---

**Description**

Create generic p\_eff function for class

**Usage**

```
p_eff(x)
```

**Arguments**

x                    An Model object.

**Value**

Numeric estimate of the number of effective parameters when computing WAIC.

**Examples**

```
m1 <- Model(ndocs = 1)
p_eff(m1)
```

---

p\_eff<- *Create generic p\_eff<- function for class*

---

**Description**

Create generic p\_eff<- function for class

**Usage**

```
p_eff(x) <- value
```

**Arguments**

x	An Model object.
value	Numeric value of effective number of parameters estimate from WAIC to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Model(ndocs = 1)
p_eff(m1) <- NaN
```

---

se\_waic *Create generic se\_waic function for class*

---

**Description**

Create generic se\_waic function for class

**Usage**

```
se_waic(x)
```

**Arguments**

x	An Model object.
---	------------------

**Value**

Numeric standard error for WAIC estimate.

**Examples**

```
m1 <- Model(ndocs = 1)
se_waic(m1)
```

---

```
se_waic<-          Create generic se_waic<- function for class
```

---

**Description**

Create generic se\_waic<- function for class

**Usage**

```
se_waic(x) <- value
```

**Arguments**

x	An Model object.
value	Numeric standard error of WAIC estimate to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Model(ndocs = 1)
se_waic(m1) <- NaN
```

---

```
sigma0          Create generic sigma0 function for class
```

---

**Description**

Create generic sigma0 function for class

**Usage**

```
sigma0(x)
```

**Arguments**

x	An Model object.
---	------------------

**Value**

Double matrix of prior variances and covariances for regression coefficients.

**Examples**

```
m1 <- Model(ndocs = 1)
sigma0(m1)
```

---

`sigma0<-` *Create generic sigma0<- function for class*

---

**Description**

Create generic `sigma0<-` function for class

**Usage**

```
sigma0(x) <- value
```

**Arguments**

`x` An `Model` object.  
`value` Numeric covariance matrix of prior for regression coefficients to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Model(ndocs = 1)
sigma0(m1) <- diag(1.0, 2)
```

---

`sigma2` *Create generic sigma2 function for class*

---

**Description**

Create generic `sigma2` function for class

**Usage**

```
sigma2(x)
```

**Arguments**

`x` An `Mlr` object.

**Value**

Numeric vector of posterior draws of residual variance.

**Examples**

```
m1 <- Mlr(ndocs = 1)
sigma2(m1)
```

---

sigma2<-	<i>Create generic sigma2&lt;- function for class</i>
----------	--

---

**Description**

Create generic sigma2<- function for class

**Usage**

```
sigma2(x) <- value
```

**Arguments**

x	An Mlr object.
value	Numeric value of residual variance to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Mlr(ndocs = 1)
sigma2(m1) <- 1.0
```

---

Sldax-class	<i>S4 class to represent a SLDAX general model that inherits from <a href="#">Mlr</a> and <a href="#">Logistic</a>.</i>
-------------	---

---

**Description**

S4 class to represent a SLDAX general model that inherits from [Mlr](#) and [Logistic](#).

Helper function (constructor) for Sldax class

**Usage**

```
## S4 method for signature 'Sldax'  
topics(x)  
  
## S4 replacement method for signature 'Sldax'  
topics(x) <- value  
  
## S4 method for signature 'Sldax'  
theta(x)  
  
## S4 replacement method for signature 'Sldax'  
theta(x) <- value  
  
## S4 method for signature 'Sldax'  
beta_(x)  
  
## S4 replacement method for signature 'Sldax'  
beta_(x) <- value  
  
## S4 method for signature 'Sldax'  
gamma_(x)  
  
## S4 replacement method for signature 'Sldax'  
gamma_(x) <- value  
  
## S4 method for signature 'Sldax'  
alpha(x)  
  
## S4 replacement method for signature 'Sldax'  
alpha(x) <- value  
  
## S4 method for signature 'Sldax'  
ntopics(x)  
  
## S4 replacement method for signature 'Sldax'  
ntopics(x) <- value  
  
## S4 method for signature 'Sldax'  
nvocab(x)  
  
## S4 replacement method for signature 'Sldax'  
nvocab(x) <- value  
  
Sldax(nvocab, topics, theta, beta, ntopics = 2, alpha = 1, gamma = 1, ...)
```

**Arguments**

x                    An Sldax object.

value	A value to assign to a slot for x
nvocab	The number of terms in the corpus vocabulary.
topics	A $D \times \max(N_d) \times M$ numeric array of topic draws. 0 indicates an unused word index (i.e., the document did not have a word at that index).
theta	A $D \times K \times M$ numeric array of topic proportions.
beta	A $K \times V \times M$ numeric array of topic-vocabulary distributions.
ntopics	The number of topics for the LDA model (default: 2).
alpha	A numeric prior hyperparameter for theta (default: $1 \cdot \theta$ ).
gamma	A numeric prior hyperparameter for beta (default: $1 \cdot \theta$ ).
...	additional arguments to be passed to the low level regression fitting functions (see below).

**Value**

A [Sldax](#) object.

**Slots**

nvocab The number of terms in the corpus vocabulary.

ntopics The number of topics for the LDA model.

alpha A numeric prior hyperparameter for theta.

gamma A numeric prior hyperparameter for beta.

topics A  $D \times \max(N_d) \times M$  numeric array of topic draws. 0 indicates an unused word index (i.e., the document did not have a word at that index).

theta A  $D \times K \times M$  numeric array of topic proportions.

beta A  $K \times V \times M$  numeric array of topic-vocabulary distributions.

**Examples**

```
m1 <- Sldax(ndocs = 1, nvocab = 2,
           topics = array(c(1, 2, 2, 1), dim = c(1, 4, 1)),
           theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
           beta = array(c(0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 1)))
nvocab(m1) <- 2L
```



**Description**

Obtain parameter estimates, model goodness-of-fit metrics, and posterior summaries.

For SLDA or SLDAX models, label switching is handled during estimation in the `gibbs_sldax()` function with argument `correct_ls`, so it is not addressed by this function.

**Usage**

```
est_beta(mcmc_fit, burn = 0, thin = 1, stat = "mean")
est_theta(mcmc_fit, burn = 0, thin = 1, stat = "mean")
get_coherence(beta_, docs, nwords = 10)
get_exclusivity(beta_, nwords = 10, weight = 0.7)
get_toptopics(theta, ntopics)
get_topwords(beta_, nwords, vocab, method = "termscore")
get_zbar(mcmc_fit, burn = 0L, thin = 1L)
post_regression(mcmc_fit)
gg_coef(mcmc_fit, burn = 0L, thin = 1L, stat = "mean", errorbw = 0.5)
## S4 method for signature 'Sldax'
gg_coef(mcmc_fit, burn = 0L, thin = 1L, stat = "mean", errorbw = 0.5)
## S4 method for signature 'Sldax'
est_beta(mcmc_fit, burn = 0, thin = 1, stat = "mean")
## S4 method for signature 'Sldax'
est_theta(mcmc_fit, burn = 0, thin = 1, stat = "mean")
## S4 method for signature 'matrix,matrix'
get_coherence(beta_, docs, nwords = 10)
## S4 method for signature 'matrix'
get_exclusivity(beta_, nwords = 10, weight = 0.7)
## S4 method for signature 'matrix'
get_toptopics(theta, ntopics)
```

```
## S4 method for signature 'matrix,numeric,character'
get_topwords(beta_, nwords, vocab, method = "termscore")
```

```
## S4 method for signature 'Sldax'
get_zbar(mcmc_fit, burn = 0L, thin = 1L)
```

```
## S4 method for signature 'Mlr'
post_regression(mcmc_fit)
```

```
## S4 method for signature 'Logistic'
post_regression(mcmc_fit)
```

```
## S4 method for signature 'Sldax'
post_regression(mcmc_fit)
```

## Arguments

<code>mcmc_fit</code>	An object of class <a href="#">Sldax</a> .
<code>burn</code>	The number of draws to discard as a burn-in period (default: 0).
<code>thin</code>	The number of draws to skip as a thinning period (default: 1; i.e., no thinning).
<code>stat</code>	The summary statistic to use on the posterior draws (default: "mean").
<code>beta_</code>	A $K \times V$ matrix of word-topic probabilities. Each row sums to 1.
<code>docs</code>	The $D \times \max(N_d)$ matrix of documents (word indices) used to fit the <a href="#">Sldax</a> model.
<code>nwords</code>	The number of words to retrieve (default: all).
<code>weight</code>	The weight (between 0 and 1) to give to exclusivity (near 1) vs. frequency (near 0). (default: 0.7).
<code>theta</code>	A $D \times K$ matrix of $K$ topic proportions for all $D$ documents.
<code>ntopics</code>	The number of topics to retrieve (default: all topics).
<code>vocab</code>	A character vector of length $V$ containing the vocabulary.
<code>method</code>	If "termscore", use term scores (similar to tf-idf). If "prob", use probabilities (default: "termscore").
<code>errorbw</code>	Positive control parameter for the width of the $\pm 2$ posterior standard error bars (default: 0.5).

## Details

- `get_zbar()` computes empirical topic proportions from slot `@topics`.
- `est_theta()` estimates the mean or median theta matrix.
- `est_beta()` estimates the mean or median beta matrix.
- `get_toptopics()` creates a [tibble](#) of the topic proportion estimates for the top `ntopics` topics per document sorted by probability.

- `get_topwords()` creates a `tibble` of topics and the top `nwords` words per topic sorted by probability or term score.
- `get_coherence()` computes the coherence metric for each topic (see Mimno, Wallach, Talley, Leenders, & McCallum, 2011).
- `get_exclusivity()` computes the exclusivity metric for each topic (see Roberts, Stewart, & Airoldi, 2013).
- `post_regression()` creates a `coda::mcmc` object containing posterior information for the regression model parameters.
- `gg_coef()` plots regression coefficients
  - Warning: this function is deprecated.
  - See `help("Deprecated")`.

## Value

A matrix of topic-word probability estimates.

A matrix of topic proportion estimates.

A numeric vector of coherence scores for each topic (more positive is better).

A numeric vector of exclusivity scores (more positive is better).

A data frame of the `ntopics` most probable topics per document.

A  $K \times V$  matrix of term-scores (comparable to tf-idf).

A matrix of empirical topic proportions per document.

An object of class `coda::mcmc` summarizing the posterior distribution of the regression coefficients and residual variance (if applicable). Convenience functions such as `summary()` and `plot()` can be used for posterior summarization.

A `ggplot` object.

## Examples

```
m1 <- Sldax(ndocs = 1, nvocab = 2,
            topics = array(c(1, 2, 2, 1), dim = c(1, 4, 1)),
            theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
            beta = array(c(0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 1)))
est_beta(m1, stat = "mean")
est_beta(m1, stat = "median")
m1 <- Sldax(ndocs = 2, nvocab = 2, nchain = 2,
            topics = array(c(1, 2, 2, 1,
                           1, 2, 2, 1), dim = c(2, 2, 2)),
            theta = array(c(0.5, 0.5,
                           0.5, 0.5,
                           0.5, 0.5,
                           0.5, 0.5), dim = c(2, 2, 2)),
            loglike = rep(NaN, times = 2),
            logpost = rep(NaN, times = 2),
            lpd = matrix(NaN, nrow = 2, ncol = 2),
            eta = matrix(0.0, nrow = 2, ncol = 2),
            mu0 = c(0.0, 0.0),
```

```

        sigma0 = diag(1, 2),
        eta_start = c(0.0, 0.0),
        beta = array(c(0.5, 0.5, 0.5, 0.5,
                      0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 2)))
est_theta(m1, stat = "mean")
est_theta(m1, stat = "median")
mdoc <- matrix(c(1, 2, 2, 1), nrow = 1)
m1 <- Sldax(ndocs = 1, nvocab = 2,
           topics = array(c(1, 2, 2, 2), dim = c(1, 4, 1)),
           theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
           beta = array(c(0.5, 0.4, 0.5, 0.6), dim = c(2, 2, 1)))
bhat <- est_beta(m1)
get_coherence(bhat, docs = mdoc, nwords = nvocab(m1))
m1 <- Sldax(ndocs = 1, nvocab = 2,
           topics = array(c(1, 2, 2, 2), dim = c(1, 4, 1)),
           theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
           beta = array(c(0.5, 0.4, 0.5, 0.6), dim = c(2, 2, 1)))
bhat <- est_beta(m1)
get_exclusivity(bhat, nwords = nvocab(m1))
m1 <- Sldax(ndocs = 2, nvocab = 2, nchain = 2,
           topics = array(c(1, 2, 2, 1,
                          1, 2, 2, 1), dim = c(2, 2, 2)),
           theta = array(c(0.4, 0.3,
                          0.6, 0.7,
                          0.45, 0.5,
                          0.55, 0.5), dim = c(2, 2, 2)),
           loglike = rep(NaN, times = 2),
           logpost = rep(NaN, times = 2),
           lpd = matrix(NaN, nrow = 2, ncol = 2),
           eta = matrix(0.0, nrow = 2, ncol = 2),
           mu0 = c(0.0, 0.0),
           sigma0 = diag(1, 2),
           eta_start = c(0.0, 0.0),
           beta = array(c(0.5, 0.5, 0.5, 0.5,
                          0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 2)))
t_hat <- est_theta(m1, stat = "mean")
get_toptopics(t_hat, ntopics = ntopics(m1))
m1 <- Sldax(ndocs = 1, nvocab = 2,
           topics = array(c(1, 2, 2, 2), dim = c(1, 4, 1)),
           theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
           beta = array(c(0.5, 0.4, 0.5, 0.6), dim = c(2, 2, 1)))
bhat <- est_beta(m1)
get_topwords(bhat, nwords = nvocab(m1), method = "termscore")
get_topwords(bhat, nwords = nvocab(m1), method = "prob")
m1 <- Sldax(ndocs = 1, nvocab = 2,
           topics = array(c(1, 2, 2, 2), dim = c(1, 4, 1)),
           theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
           beta = array(c(0.5, 0.4, 0.5, 0.6), dim = c(2, 2, 1)))
get_zbar(m1)
data(mtcars)
m1 <- gibbs_mlr(mpg ~ hp, data = mtcars, m = 2)
post_regression(m1)
## Not run:

```

```

library(lda) # Required if using `prep_docs()`
data(teacher_rate) # Synthetic student ratings of instructors
docs_vocab <- prep_docs(teacher_rate, "doc")
vocab_len <- length(docs_vocab$vocab)
m1 <- gibbs_sldax(rating ~ I(grade - 1), m = 2,
                 data = teacher_rate,
                 docs = docs_vocab$documents,
                 V = vocab_len,
                 K = 2,
                 model = "sldax")

gg_coef(m1)

## End(Not run)

```

---

teacher_rate	<i>Synthetic (fake) student ratings of instructor quality.</i>
--------------	--

---

### Description

A data set containing almost 3,800 student ratings and written comments regarding instructor quality along with the students' grades associated with the course.

### Usage

```
teacher_rate
```

### Format

A data frame with 3,733 rows and 4 variables:

**id** Row number to identify rater

**rating** A numerical rating of instructor quality from 1 (worst) to 5 (best)

**grade** A numerical grade received by the rater for the instructor's course ranging from 1 (worst) to 13 (best)

**doc** A character vector containing pseudo-written comments about the instructors

---

term_score	<i>Compute term-scores for each word-topic pair</i>
------------	---

---

### Description

For more details, see Blei, D. M., & Lafferty, J. D. (2009). Topic models. In A. N. Srivastava & M. Sahami (Eds.), Text mining: Classification, clustering, and applications. Chapman and Hall/CRC.

### Usage

```
term_score(beta_)
```

**Arguments**

beta\_                    A  $K \times V$  matrix of  $V$  vocabulary probabilities for each of  $K$  topics.

**Value**

A  $K \times V$  matrix of term-scores (comparable to tf-idf).

**Examples**

```
#' library(lda) # Required if using `prep_docs()`

data(teacher_rate) # Synthetic student ratings of instructors
docs_vocab <- prep_docs(teacher_rate, "doc")
vocab_len <- length(docs_vocab$vocab)
m1 <- gibbs_sldax(rating ~ I(grade - 1), m = 2,
                 data = teacher_rate, docs = docs_vocab$documents,
                 V = vocab_len, K = 2, model = "sldax")
hbeta <- est_beta(m1)
ts_beta <- term_score(hbeta)
# One row per topic, one column per unique term in the vocabulary
str(ts_beta)
```

---

theta

*Create generic theta function for class*


---

**Description**

Create generic theta function for class

**Usage**

```
theta(x)
```

**Arguments**

x                        An Sldax object.

**Value**

Numeric array of topic proportions for each document across sampler iterations.

**Examples**

```
m1 <- Sldax(ndocs = 1, nvocab = 2,
            topics = array(c(1, 2, 2, 1), dim = c(1, 4, 1)),
            theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
            beta = array(c(0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 1)))
theta(m1)
```

---

theta<-                      *Create generic theta<- function for class*

---

**Description**

Create generic theta<- function for class

**Usage**

```
theta(x) <- value
```

**Arguments**

x	An Sldax object.
value	Numeric array of topic proportions to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Sldax(ndocs = 1, nvocab = 2,
           topics = array(c(1, 2, 2, 1), dim = c(1, 4, 1)),
           theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
           beta = array(c(0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 1)))
theta(m1) <- array(c(0.5, 0.5), dim = c(1, 2, 1))
```

---

topics                      *Create generic topics function for class*

---

**Description**

Create generic topics function for class

**Usage**

```
topics(x)
```

**Arguments**

x	An Sldax object.
---	------------------

**Value**

Integer array of categorical topic labels for each word in each document across sampler iterations.

**Examples**

```
m1 <- Sldax(ndocs = 1, nvocab = 2,
           topics = array(c(1, 2, 2, 1), dim = c(1, 4, 1)),
           theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
           beta = array(c(0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 1)))
topics(m1)
```

---

topics<- *Create generic topics<- function for class*

---

**Description**

Create generic topics<- function for class

**Usage**

```
topics(x) <- value
```

**Arguments**

x                    An Sldax object.  
value                Integer array of topic assignment draws for each word to assign to slot.

**Value**

None.

**Examples**

```
m1 <- Sldax(ndocs = 1, nvocab = 2,
           topics = array(c(1, 2, 2, 1), dim = c(1, 4, 1)),
           theta = array(c(0.5, 0.5), dim = c(1, 2, 1)),
           beta = array(c(0.5, 0.5, 0.5, 0.5), dim = c(2, 2, 1)))
topics(m1) <- array(c(2, 2, 2, 1), dim = c(1, 4, 1))
```

---

waic *Create generic waic function for class*

---

**Description**

Create generic waic function for class

**Usage**

```
waic(x)
```



`waic<-`

49

### Arguments

`x`                    An Model object.

### Value

Numeric value of the Watanabe Information Criterion (WAIC).

### Examples

```
m1 <- Model(ndocs = 1)
waic(m1)
```

---

`waic<-`                    *Create generic waic<- function for class*

---

### Description

Create generic `waic<-` function for class

### Usage

```
waic(x) <- value
```

### Arguments

`x`                    An Model object.  
`value`                Numeric value of WAIC to assign to slot.

### Value

None.

### Examples

```
m1 <- Model(ndocs = 1)
waic(m1) <- NaN
```

---

waic_all	<i>Compute WAIC for all outcomes.</i>
----------	---------------------------------------

---

**Description**

Compute WAIC for all outcomes.

**Usage**

```
waic_all(iter, l_pred)
```

**Arguments**

iter	The length of the sampled chain.
l_pred	A iter x D matrix of predictive likelihoods (NOT log-likelihoods).

**Value**

Vector of (1) WAIC for model, (2) standard error for WAIC, and (3) the effective number of parameters.

**Examples**

```
data(teacher_rate)
fit_mlr <- gibbs_mlr(rating ~ grade, data = teacher_rate, m = 5)
waic_all(iter = 5, t(lpd(fit_mlr)))
```

---

waic_d	<i>WAIC for observation y_d</i>
--------	---------------------------------

---

**Description**

WAIC for observation y\_d

**Arguments**

like_pred	A m x 1 vector of predictive likelihoods (NOT log-likelihoods) for y_d.
p_effd	The contribution to the effective number of parameters from obs y_d.

**Value**

WAIC contribution for observation d (on deviance scale).

---

waic_diff	<i>Compute difference (WAIC1 - WAIC2) in WAIC and its SE for two models.</i>
-----------	--

---

**Description**

Compute difference (WAIC1 - WAIC2) in WAIC and its SE for two models.

**Usage**

```
waic_diff(l_pred1, l_pred2)
```

**Arguments**

l_pred1	A $m_1 \times D$ matrix of predictive likelihoods (NOT log-likelihoods) from model 1.
l_pred2	A $m_2 \times D$ matrix of predictive likelihoods (NOT log-likelihoods) from model 2.

**Value**

A vector of (1) the difference in WAIC (on the deviance scale) between models and (2) the standard error of the difference in WAIC.

**Examples**

```
data(teacher_rate)
fit_mlr <- gibbs_mlr(rating ~ grade, data = teacher_rate, m = 100)
fit_mlr2 <- gibbs_mlr(rating ~ grade + I(grade^2), data = teacher_rate, m = 100)
# Returns (1) D = WAIC(fit_mlr2) - WAIC(fit_mlr) and (2) SE(D)
# Suggests that a linear relationship is preferable
waic_diff(t(lpd(fit_mlr2)), t(lpd(fit_mlr)))
```

# Index

- \* **Gibbs sampler**
  - [gibbs\\_logistic](#), [12](#)
  - [gibbs\\_mlr](#), [14](#)
  - [gibbs\\_sldax](#), [15](#)
- \* **classes**
  - [Logistic-class](#), [18](#)
  - [Mlr-class](#), [22](#)
  - [Model-class](#), [23](#)
  - [Sldax-class](#), [38](#)
- \* **datasets**
  - [teacher\\_rate](#), [45](#)
- [a0](#), [3](#)
- [a0](#), [Mlr-method \(Mlr-class\)](#), [22](#)
- [a0<-](#), [4](#)
- [a0<-](#), [Mlr-method \(Mlr-class\)](#), [22](#)
- [alpha](#), [4](#)
- [alpha](#), [Sldax-method \(Sldax-class\)](#), [38](#)
- [alpha<-](#), [5](#)
- [alpha<-](#), [Sldax-method \(Sldax-class\)](#), [38](#)
- [b0](#), [6](#)
- [b0](#), [Mlr-method \(Mlr-class\)](#), [22](#)
- [b0<-](#), [6](#)
- [b0<-](#), [Mlr-method \(Mlr-class\)](#), [22](#)
- [beta\\_](#), [7](#)
- [beta\\_](#), [Sldax-method \(Sldax-class\)](#), [38](#)
- [beta\\_<-](#), [7](#)
- [beta\\_<-](#), [Sldax-method \(Sldax-class\)](#), [38](#)
- [coda::mcmc](#), [43](#)
- [est\\_beta \(sldax-summary\)](#), [41](#)
- [est\\_beta](#), [Sldax-method \(sldax-summary\)](#), [41](#)
- [est\\_theta \(sldax-summary\)](#), [41](#)
- [est\\_theta](#), [Sldax-method \(sldax-summary\)](#), [41](#)
- [eta](#), [8](#)
- [eta](#), [Model-method \(Model-class\)](#), [23](#)
- [eta<-](#), [8](#)
- [eta<-](#), [Model-method \(Model-class\)](#), [23](#)
- [eta\\_start](#), [9](#)
- [eta\\_start](#), [Model-method \(Model-class\)](#), [23](#)
- [eta\\_start<-](#), [9](#)
- [eta\\_start<-](#), [Model-method \(Model-class\)](#), [23](#)
- [extra](#), [10](#)
- [extra](#), [Model-method \(Model-class\)](#), [23](#)
- [extra<-](#), [10](#)
- [extra<-](#), [Model-method \(Model-class\)](#), [23](#)
- [formula](#), [13](#), [14](#), [16](#)
- [gamma\\_](#), [11](#)
- [gamma\\_](#), [Sldax-method \(Sldax-class\)](#), [38](#)
- [gamma\\_<-](#), [12](#)
- [gamma\\_<-](#), [Sldax-method \(Sldax-class\)](#), [38](#)
- [get\\_coherence \(sldax-summary\)](#), [41](#)
- [get\\_coherence](#), [matrix](#), [matrix-method \(sldax-summary\)](#), [41](#)
- [get\\_exclusivity \(sldax-summary\)](#), [41](#)
- [get\\_exclusivity](#), [matrix-method \(sldax-summary\)](#), [41](#)
- [get\\_toptopics \(sldax-summary\)](#), [41](#)
- [get\\_toptopics](#), [matrix-method \(sldax-summary\)](#), [41](#)
- [get\\_topwords \(sldax-summary\)](#), [41](#)
- [get\\_topwords](#), [matrix](#), [numeric](#), [character-method \(sldax-summary\)](#), [41](#)
- [get\\_zbar \(sldax-summary\)](#), [41](#)
- [get\\_zbar](#), [Sldax-method \(sldax-summary\)](#), [41](#)
- [gg\\_coef \(sldax-summary\)](#), [41](#)
- [gg\\_coef](#), [Sldax-method \(sldax-summary\)](#), [41](#)
- [gibbs\\_logistic](#), [12](#), [15](#), [17](#)
- [gibbs\\_logistic\(\)](#), [34](#)
- [gibbs\\_mlr](#), [13](#), [14](#), [17](#)
- [gibbs\\_mlr\(\)](#), [34](#)
- [gibbs\\_sldax](#), [13](#), [15](#), [15](#)

- `gibbs_sldax()`, [34](#), [41](#)
- Logistic, [13](#), [18](#), [38](#)
- Logistic (Logistic-class), [18](#)
- Logistic-class, [18](#)
- loglike, [19](#)
- loglike, Model-method (Model-class), [23](#)
- loglike<-, [19](#)
- loglike<-, Model-method (Model-class), [23](#)
- logpost, [20](#)
- logpost, Model-method (Model-class), [23](#)
- logpost<-, [20](#)
- logpost<-, Model-method (Model-class), [23](#)
- lpd, [21](#)
- lpd, Model-method (Model-class), [23](#)
- lpd<-, [21](#)
- lpd<-, Model-method (Model-class), [23](#)
- Mlr, [15](#), [23](#), [38](#)
- Mlr (Mlr-class), [22](#)
- Mlr-class, [22](#)
- Model, [18](#), [22](#), [25](#)
- Model (Model-class), [23](#)
- Model-class, [23](#)
- $\mu_0$ , [26](#)
- $\mu_0$ , Model-method (Model-class), [23](#)
- $\mu_0$ <-, [27](#)
- $\mu_0$ <-, Model-method (Model-class), [23](#)
- nchain, [27](#)
- nchain, Model-method (Model-class), [23](#)
- nchain<-, [28](#)
- nchain<-, Model-method (Model-class), [23](#)
- ndocs, [28](#)
- ndocs, Model-method (Model-class), [23](#)
- ndocs<-, [29](#)
- ndocs<-, Model-method (Model-class), [23](#)
- ntopics, [29](#)
- ntopics, Sldax-method (Sldax-class), [38](#)
- ntopics<-, [30](#)
- ntopics<-, Sldax-method (Sldax-class), [38](#)
- nvocab, [30](#)
- nvocab, Sldax-method (Sldax-class), [38](#)
- nvocab<-, [31](#)
- nvocab<-, Sldax-method (Sldax-class), [38](#)
- $p_{\text{eff}}$ , [34](#)
- $p_{\text{eff}}$ , Model-method (Model-class), [23](#)
- $p_{\text{eff}}$ <-, [35](#)
- $p_{\text{eff}}$ <-, Model-method (Model-class), [23](#)
- plot(), [43](#)
- post\_regression (sldax-summary), [41](#)
- post\_regression, Logistic-method (sldax-summary), [41](#)
- post\_regression, Mlr-method (sldax-summary), [41](#)
- post\_regression, Sldax-method (sldax-summary), [41](#)
- prep\_docs, [32](#)
- proposal\_sd, [33](#)
- proposal\_sd, Logistic-method (Logistic-class), [18](#)
- proposal\_sd<-, [33](#)
- proposal\_sd<-, Logistic-method (Logistic-class), [18](#)
- psychtm, [34](#)
- se\_waic, [35](#)
- se\_waic, Model-method (Model-class), [23](#)
- se\_waic<-, [36](#)
- se\_waic<-, Model-method (Model-class), [23](#)
- $\sigma_0$ , [36](#)
- $\sigma_0$ , Model-method (Model-class), [23](#)
- $\sigma_0$ <-, [37](#)
- $\sigma_0$ <-, Model-method (Model-class), [23](#)
- $\sigma_2$ , [37](#)
- $\sigma_2$ , Mlr-method (Mlr-class), [22](#)
- $\sigma_2$ <-, [38](#)
- $\sigma_2$ <-, Mlr-method (Mlr-class), [22](#)
- Sldax, [17](#), [40–42](#)
- Sldax (Sldax-class), [38](#)
- Sldax-class, [38](#)
- sldax-summary, [41](#)
- summary(), [43](#)
- teacher\_rate, [45](#)
- term\_score, [45](#)
- theta, [46](#)
- theta, Sldax-method (Sldax-class), [38](#)
- theta<-, [47](#)
- theta<-, Sldax-method (Sldax-class), [38](#)
- tibble, [42](#), [43](#)
- topics, [47](#)
- topics, Sldax-method (Sldax-class), [38](#)
- topics<-, [48](#)
- topics<-, Sldax-method (Sldax-class), [38](#)
- waic, [48](#)

`waic`, Model-method (Model-class), [23](#)  
`waic<-`, [49](#)  
`waic<-`, Model-method (Model-class), [23](#)  
`waic_all`, [50](#)  
`waic_d`, [50](#)  
`waic_diff`, [51](#)