Package ‘LDATS’

March 19, 2020

Title  Latent Dirichlet Allocation Coupled with Time Series Analyses

Version 0.2.7


URL https://weecology.github.io/LDATS,
    https://github.com/weecology/LDATS

BugReports https://github.com/weecology/LDATS/issues

Depends R (>= 3.2.3)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports coda, digest, extraDistr, graphics, grDevices, lubridate, magrittr, memoise, methods, mvtnorm, nnet, progress, stats, topicmodels, viridis

Suggests knitr, pkgdown, rmarkdown, testthat, vdiffr

VignetteBuilder knitr

RoxygenNote 6.1.1

NeedsCompilation no

Author Juniper L. Simonis [aut, cre] (<https://orcid.org/0000-0001-9798-0460>),
    Erica M. Christensen [aut] (<https://orcid.org/0000-0002-5635-2502>),
    David J. Harris [aut] (<https://orcid.org/0000-0003-3332-9307>),
    Renata M. Diaz [aut] (<https://orcid.org/0000-0003-0803-4734>),
    Hao Ye [aut] (<https://orcid.org/0000-0002-8630-1458>)
Ethan P. White [aut] (<https://orcid.org/0000-0001-6728-7745>),
S.K. Morgan Ernest [aut] (<https://orcid.org/0000-0002-6026-8530>),
Weecology [cph]

Maintainer Juniper L. Simonis <juniper.simonis@weecology.org>
Repository CRAN
Date/Publication 2020-03-19 11:10:02 UTC

R topics documented:

AICc .................................................. 4
autocorr_plot .................................... 4
check_changepoints ............................. 5
check_control .................................... 5
check_document_covariate_table .......... 6
check_document_term_table ............... 7
check_formula .................................. 7
check_formulas ................................ 8
check_LDA_models ............................ 9
check_nchangepoints ........................ 10
check_seeds ................................... 10
check_timename ............................... 11
check_topics .................................. 11
check_weights ................................. 12
count_trips ................................... 13
diagnose_ptMCMC ............................... 14
document_weights ............................ 15
edcf_plot ....................................... 16
est_changepoints .............................. 16
est_regressors ................................. 18
expand_TS ..................................... 20
iftrue ......................................... 21
jornada ......................................... 21
LDATS ........................................... 22
LDA_msg ......................................... 23
LDA_set ......................................... 23
LDA_set_control ............................... 24
LDA_TS .......................................... 25
LDA_TS_control ............................... 27
logLik.LDA_VEM ............................... 29
logLik.multinom_TS_fit ..................... 30
logLik.TS_fit ................................ 30
logsumexp ...................................... 31
memoise_fun .................................. 32
messageq ...................................... 32
mirror_vcov ................................... 33
modalvalue .................................... 33
multinom_TS ................................... 34
topics documented:

multinom_TS_chunk ................................................................. 36
normalize ................................................................. 37
package_chunk_fits ................................................................. 37
package_LDA_set ................................................................. 38
package_LDA_TS ................................................................. 39
package_TS ................................................................. 40
package_TS_on_LDA ................................................................. 42
plot.LDA_set ................................................................. 43
plot.LDA_TS ................................................................. 44
plot.LDA_VEM ................................................................. 45
plot.TS_fit ................................................................. 46
posterior_plot ................................................................. 48
prep_chunks ................................................................. 48
prep_cpts ................................................................. 49
prep_ids ................................................................. 51
prep_LDA_control ................................................................. 52
prep_pbar ................................................................. 53
prep_proposal_dist ................................................................. 54
prep_ptMCMC_inputs ................................................................. 55
prep_saves ................................................................. 56
prep_temp_sequence ................................................................. 57
prep_TS_data ................................................................. 58
print.LDA_TS ................................................................. 59
print.TS_fit ................................................................. 60
print.TS_on_LDA ................................................................. 61
print_model_run_message ................................................................. 61
proposed_step_mods ................................................................. 62
rho_lines ................................................................. 63
rodents ................................................................. 64
select_LDA ................................................................. 65
select_TS ................................................................. 66
set_gamma_colors ................................................................. 67
set_LDA_plot_colors ................................................................. 68
set_LDA_TS_plot_cols ................................................................. 69
set_rho_hist_colors ................................................................. 70
set_TS_summary_plot_cols ................................................................. 71
sim_LDA_data ................................................................. 72
sim_LDA_TS_data ................................................................. 73
sim_TS_data ................................................................. 74
softmax ................................................................. 75
step_chains ................................................................. 75
summarize_etas ................................................................. 77
summarize_rhos ................................................................. 78
swap_chains ................................................................. 79
trace_plot ................................................................. 80
TS ................................................................. 81
TS_control ................................................................. 84
TS_diagnostics_plot ................................................................. 85
### AICc

**Description**

Calculate the small sample size correction of AIC for the input object.

**Usage**

```r
AICc(object)
```

**Arguments**

- `object` An object for which `AIC` and `logLik` have defined methods.

**Value**

numeric value of AICc.

**Examples**

```r
data <- data.frame(y = rnorm(50), x = rnorm(50))
mod <- lm(data)
AICc(mod)
```

### autocorr_plot

**Description**

Produce a vanilla ACF plot using `acf` for the parameter of interest (rho or eta) as part of `TS_diagnostics_plot`.

**Usage**

```r
autocorr_plot(x)
```

**Arguments**

- `x` Vector of parameter values drawn from the posterior distribution, indexed to the iteration by the order of the vector.
**check_changepoints**

Value

NULL.

Examples

```
autocorr_plot(rnorm(100, 0, 1))
```

---

**check_changepoints**  
*Check that a set of change point locations is proper*

**Description**

Check that the change point locations are numeric and conformable to integer values.

**Usage**

```
check_changepoints(changepoints = NULL)
```

**Arguments**

- **changepoints**: Change point locations to evaluate.

**Value**

An error message is thrown if changepoints are not proper, else NULL.

**Examples**

```
check_changepoints(100)
```

---

**check_control**  
*Check that a control list is proper*

**Description**

Check that a list of controls is of the right class.

**Usage**

```
check_control(control, eclass = "list")
```

**Arguments**

- **control**: Control list to evaluate.
- **eclass**: Expected class of the list to be evaluated.
Value

an error message is thrown if the input is improper, otherwise NULL.

Examples

check_control(list())

check_document_covariate_table

Check that the document covariate table is proper

Description

Check that the table of document-level covariates is conformable to a data frame and of the right size (correct number of documents) for the document-topic output from the LDA models.

Usage

check_document_covariate_table(document_covariate_table, LDA_models = NULL, document_term_table = NULL)

Arguments

document_covariate_table
  Document covariate table to evaluate.

LDA_models
  Reference LDA model list (class LDA_set) that includes as its first element a properly fitted LDA model with a gamma slot with the document-topic distribution.

document_term_table
  Optional input for checking when LDA_models is NULL

Value

An error message is thrown if document_covariate_table is not proper, else NULL.

Examples

data(rodents)
check_document_covariate_table(rodents$document_covariate_table)
check_document_term_table

Check that document term table is proper

Description

Check that the table of observations is conformable to a matrix of integers.

Usage

check_document_term_table(document_term_table)

Arguments

document_term_table
  Table of observation count data (rows: documents, columns: terms. May be a
class matrix or data.frame but must be conformable to a matrix of integers,
as verified by check_document_term_table.

Value

an error message is thrown if the input is improper, otherwise NULL.

Examples

data(rodents)
check_document_term_table(rodents$document_term_table)

check_formula

Check that a formula is proper

Description

Check that formula is actually a formula and that the response and predictor variables are all
included in data.

Usage

check_formula(data, formula)
Arguments

data frame including [1] the time variable (indicated in timename), [2] the predictor variables (required by formula) and [3], the multinomial response variable (indicated in formula) as verified by check_timename and check_formula. Note that the response variables should be formatted as a data.frame object named as indicated by the response entry in the control list, such as gamma for a standard TS analysis on LDA output.

formula formula to evaluate.

Value

An error message is thrown if formula is not proper, else NULL.

Examples

data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models$gamma
check_formula(data, gamma ~ 1)
check_LDA_models

Value

An error message is thrown if formulas is not proper, else NULL.

Examples

data(rodents)
check_formulas(~ 1, rodents$document_covariate_table)

check_LDA_models  Check that LDA model input is proper

Description

Check that the LDA_models input is either a set of LDA models (class LDA_set, produced by
LDA_set) or a singular LDA model (class LDA, produced by LDA).

Usage

check_LDA_models(LDA_models)

Arguments

LDA_models  List of LDA models or singular LDA model to evaluate.

Value

An error message is thrown if LDA_models is not proper, else NULL.

Examples

data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDAs <- LDA_set(document_term_table, topics = 2, nseeds = 1)
LDA_models <- select_LDA(LDAs)
check_LDA_models(LDA_models)
check_nchangepoints  
*Check that nchangepoints vector is proper*

**Description**

Check that the vector of numbers of changepoints is conformable to integers greater than 1.

**Usage**

```
check_nchangepoints(nchangepoints)
```

**Arguments**

- `nchangepoints`: Vector of the number of changepoints to evaluate.

**Value**

An error message is thrown if `nchangepoints` is not proper, else `NULL`.

**Examples**

```
check_nchangepoints(0)
check_nchangepoints(2)
```

check_seeds  
*Check that nseeds value or seeds vector is proper*

**Description**

Check that the vector of numbers of seeds is conformable to integers greater than 0.

**Usage**

```
check_seeds(nseeds)
```

**Arguments**

- `nseeds`: Integer number of seeds (replicate starts) to use for each value of topics in the LDAs. Must be conformable to a positive integer value.

**Value**

An error message is thrown if the input is improper, otherwise `NULL`. 
check_timename

Examples

check_seeds(1)
check_seeds(2)

check_timename  Check that the time vector is proper

Description

Check that the vector of time values is included in the document covariate table and that it is either an integer-conformable or a date. If it is a date, the input is converted to an integer, resulting in the timestep being 1 day, which is often not desired behavior.

Usage

check_timename(document_covariate_table, timename)

Arguments

document_covariate_table  Document covariate table used to query for the time column.
timename  Column name for the time variable to evaluate.

Value

An error message is thrown if timename is not proper, else NULL.

Examples

data(rodents)
check_timename(rodents$document_covariate_table, "newmoon")

check_topics  Check that topics vector is proper

Description

Check that the vector of numbers of topics is conformable to integers greater than 1.

Usage

check_topics(topics)
check_weights

Arguments

  topics  Vector of the number of topics to evaluate for each model. Must be conformable to integer values.

Value

  an error message is thrown if the input is improper, otherwise NULL.

Examples

  check_topics(2)

check_weights  Check that weights vector is proper

Description

  Check that the vector of document weights is numeric and positive and inform the user if the average weight isn’t 1.

Usage

  check_weights(weights)

Arguments

  weights  Vector of the document weights to evaluate, or TRUE for triggering internal weighting by document sizes.

Value

  An error message is thrown if weights is not proper, else NULL.

Examples

  check_weights(1)
  wts <- runif(100, 0.1, 100)
  check_weights(wts)
  wts2 <- wts / mean(wts)
  check_weights(wts2)
  check_weights(TRUE)
count_trips

**Count trips of the ptMCMC particles**

**Description**

Count the full trips (from one extreme temperature chain to the other and back again; Katzgraber *et al.* 2006) for each of the ptMCMC particles, as identified by their id on initialization.

This function was designed to work within TS and process the output of `est_changepoints` as a component of `diagnose_ptMCMC`, but has been generalized and would work with any output from a ptMCMC as long as ids is formatted properly.

**Usage**

```r
count_trips(ids)
```

**Arguments**

- `ids` matrix of identifiers of the particles in each chain for each iteration of the ptMCMC algorithm (rows: chains, columns: iterations).

**Value**


**References**


**Examples**

```r
data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)$[1]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)
data <- data[order(data[,"newmoon"]), ]
rho_dist <- est_changepoints(data, gamma ~ 1, 1, "newmoon", weights,
                          TS_control())
count_trips(rho_dist$ids)
```
**diagnose_ptMCMC**

*Calculate ptMCMC summary diagnostics*

**Description**

Summarize the step and swap acceptance rates as well as trip metrics from the saved output of a ptMCMC estimation.

**Usage**

```r
diagnose_ptMCMC(ptMCMCout)
```

**Arguments**

- `ptMCMCout`: Named list of saved data objects from a ptMCMC estimation including elements named `step_accepts` (matrix of logical outcomes of each step; rows: chains, columns: iterations), `swap_accepts` (matrix of logical outcomes of each swap; rows: chain pairs, columns: iterations), and `ids` (matrix of particle identifiers; rows: chains, columns: iterations). `ptMCMCout = NULL` indicates no use of ptMCMC and so the function returns NULL.

**Details**

Within-chain step acceptance rates are averaged for each of the chains from the raw step acceptance histories (`ptMCMCout$step_accepts`) and between-chain swap acceptance rates are similarly averaged for each of the neighboring pairs of chains from the raw swap acceptance histories (`ptMCMCout$swap_accepts`). Trips are defined as movement from one extreme chain to the other and back again (Katzgraber et al. 2006). Trips are counted and turned to per-iteration rates using `count_trips`.

This function was first designed to work within TS and process the output of `est_changepoints`, but has been generalized and would work with any output from a ptMCMC as long as `ptMCMCout` is formatted properly.

**Value**


**References**

**document_weights**

Calculate document weights for a corpus

### Description

Simple calculation of document weights based on the average number of words in a document within the corpus (mean value = 1).

### Usage

```r
document_weights(document_term_table)
```

### Arguments

- **document_term_table**
  
  Table of observation count data (rows: documents, columns: terms). May be a class matrix or data.frame but must be conformable to a matrix of integers, as verified by `check_document_term_table`.

### Value

Vector of weights, one for each document, with the average sample receiving a weight of 1.0.

### Examples

```r
data(rodents)
document_weights(rodents$document_term_table)
```
ecdf_plot Produce the posterior distribution ECDF panel for the TS diagnostic plot of a parameter

Description

Produce a vanilla ECDF (empirical cumulative distribution function) plot using ecdf for the parameter of interest (rho or eta) as part of TS_diagnostics_plot. A horizontal line is added to show the median of the posterior.

Usage

ecdf_plot(x, xlab = "parameter value")

Arguments

x Vector of parameter values drawn from the posterior distribution, indexed to the iteration by the order of the vector.

xlab character value used to label the x axis.

Value

NULL.

Examples

ecdf_plot(rnorm(100, 0, 1))

est_changepoints Use ptMCMC to estimate the distribution of change point locations

Description

This function executes ptMCMC-based estimation of the change point location distributions for multinomial Time Series analyses.

Usage

est_changepoints(data, formula, nchangepoints, timename, weights, control = list())
**est_changepoints**

**Arguments**

- **data** `data.frame` including [1] the time variable (indicated in `timename`), [2] the predictor variables (required by `formula`) and [3], the multinomial response variable (indicated in `formula`) as verified by `check_timename` and `check_formula`. Note that the response variables should be formatted as a `data.frame` object named as indicated by the response entry in the `control` list, such as `gamma` for a standard TS analysis on LDA output.

- **formula** `formula` defining the regression between relationship the change points. Any predictor variable included must also be a column in `data` and any (multinomial) response variable must be a set of columns in `data`, as verified by `check_formula`.

- **nchangepoints** integer corresponding to the number of change points to include in the model. 0 is a valid input (corresponding to no change points, so a singular time series model), and the current implementation can reasonably include up to 6 change points. The number of change points is used to dictate the segmentation of the time series into chunks fit with separate models dictated by `formula`.

- **timename** character element indicating the time variable used in the time series.

- **weights** Optional class numeric vector of weights for each document. Defaults to NULL, translating to an equal weight for each document. When using `multinom_TS` in a standard LDATS analysis, it is advisable to weight the documents by their total size, as the result of `LDA` is a matrix of proportions, which does not account for size differences among documents. For most models, a scaling of the weights (so that the average is 1) is most appropriate, and this is accomplished using `document_weights`.

- **control** A list of parameters to control the fitting of the Time Series model including the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values not input assume defaults set by `TS_control`.

**Value**

List of saved data objects from the ptMCMC estimation of change point locations (unless `nchangepoints` is 0, then NULL is returned).

**Examples**

```r
data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)
formula <- gamma ~ 1
nchangepoints <- 1
control <- TS_control()
data <- data[order(data[, "newmoon"]), ]
rho_dist <- est_changepoints(data, formula, nchangepoints, "newmoon",
```

```r
```
```
**est_regressors**

Estimate the distribution of regressors, unconditional on the change point locations

**Description**

This function uses the marginal posterior distributions of the change point locations (estimated by `est_changepoints`) in combination with the conditional (on the change point locations) posterior distributions of the regressors (estimated by `multinom_TS`) to estimate the marginal posterior distribution of the regressors, unconditional on the change point locations.

**Usage**

```r
est_regressors(rho_dist, data, formula, timename, weights, control = list())
```

**Arguments**

- **rho_dist**: List of saved data objects from the ptMCMC estimation of change point locations (unless `nchangepoints` is 0, then NULL) returned from `est_changepoints`.
- **data**: `data.frame` including [1] the time variable (indicated in `timename`), [2] the predictor variables (required by `formula`) and [3], the multinomial response variable (indicated in `formula`) as verified by `check_timename` and `check_formula`. Note that the response variables should be formatted as a `data.frame` object named as indicated by the response entry in the control list, such as `gamma` for a standard TS analysis on LDA output.
- **formula**: `formula` defining the regression between relationship the change points. Any predictor variable included must also be a column in data and any (multinomial) response variable must be a set of columns in data, as verified by `check_formula`.
- **timename**: character element indicating the time variable used in the time series.
- **weights**: Optional class numeric vector of weights for each document. Defaults to NULL, translating to an equal weight for each document. When using `multinom_TS` in a standard LDATS analysis, it is advisable to weight the documents by their total size, as the result of `LDA` is a matrix of proportions, which does not account for size differences among documents. For most models, a scaling of the weights (so that the average is 1) is most appropriate, and this is accomplished using `document_weights`.
- **control**: A list of parameters to control the fitting of the Time Series model including the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values not input assume defaults set by `TS_control`.

---

**Arguments**

- **rho_dist**: List of saved data objects from the ptMCMC estimation of change point locations (unless `nchangepoints` is 0, then NULL) returned from `est_changepoints`.
- **data**: `data.frame` including [1] the time variable (indicated in `timename`), [2] the predictor variables (required by `formula`) and [3], the multinomial response variable (indicated in `formula`) as verified by `check_timename` and `check_formula`. Note that the response variables should be formatted as a `data.frame` object named as indicated by the response entry in the control list, such as `gamma` for a standard TS analysis on LDA output.
- **formula**: `formula` defining the regression between relationship the change points. Any predictor variable included must also be a column in data and any (multinomial) response variable must be a set of columns in data, as verified by `check_formula`.
- **timename**: character element indicating the time variable used in the time series.
- **weights**: Optional class numeric vector of weights for each document. Defaults to NULL, translating to an equal weight for each document. When using `multinom_TS` in a standard LDATS analysis, it is advisable to weight the documents by their total size, as the result of `LDA` is a matrix of proportions, which does not account for size differences among documents. For most models, a scaling of the weights (so that the average is 1) is most appropriate, and this is accomplished using `document_weights`.
- **control**: A list of parameters to control the fitting of the Time Series model including the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values not input assume defaults set by `TS_control`. 
Details

The general approach follows that of Western and Kleykamp (2004), although we note some important differences. Our regression models are fit independently for each chunk (segment of time), and therefore the variance-covariance matrix for the full model has 0 entries for covariances between regressors in different chunks of the time series. Further, because the regression model here is a standard (non-hierarchical) softmax (Ripley 1996, Venables and Ripley 2002, Bishop 2006), there is no error term in the regression (as there is in the normal model used by Western and Kleykamp 2004), and so the posterior distribution used here is a multivariate normal, as opposed to a multivariate t, as used by Western and Kleykamp (2004).

Value

matrix of draws (rows) from the marginal posteriors of the coefficients across the segments (columns).

References


Examples

data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)
formula <- gamma ~ 1
nchangepoints <- 1
control <- TS_control()
data <- data[order(data[, "newmoon"], )]
rho_dist <- est_changepoints(data, formula, nchangepoints, "newmoon", weights, control)
eta_dist <- est_regressors(rho_dist, data, formula, "newmoon", weights, control)
Expand the TS models across the factorial combination of LDA models, formulas, and number of change points

**Description**
Expand the completely crossed combination of model inputs: LDA model results, formulas, and number of change points.

**Usage**
expand_TS(LDA_models, formulas, nchangepoints)

**Arguments**
- LDA_models: List of LDA models (class LDA_set, produced by LDA_set) or a singular LDA model (class LDA, produced by LDA).
- formulas: Vector of formula(s) for the continuous (non-change point) component of the time series models. Any predictor variable included in a formula must also be a column in the document_covariate_table. Each element (formula) in the vector is evaluated for each number of change points and each LDA model.
- nchangepoints: Vector of integers corresponding to the number of change points to include in the time series models. 0 is a valid input corresponding to no change points (i.e., a singular time series model), and the current implementation can reasonably include up to 6 change points. Each element in the vector is the number of change points used to segment the data for each formula (entry in formulas) component of the TS model, for each selected LDA model.

**Value**
Expanded data.frame table of the three values (columns) for each unique model run (rows): [1] the LDA model (indicated as a numeric element reference to the LDA_models object), [2] the regressor formula, and [3] the number of changepoints.

**Examples**
```
data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDAs <- LDA_set(document_term_table, topics = 2:3, nseeds = 2)
LDA_models <- select_LDA(LDAs)
weights <- document_weights(document_term_table)
formulas <- c(~ 1, ~ newmoon)
nchangepoints <- 0:1
expand_TS(LDA_models, formulas, nchangepoints)
```
**iftrue**

*Replace if TRUE*

**Description**

If the focal input is TRUE, replace it with alternative.

**Usage**

```r
iftrue(x = TRUE, alt = NULL)
```

**Arguments**

- `x` Focal input.
- `alt` Alternative value.

**Value**

`x` if not TRUE, `alt` otherwise.

**Examples**

```r
code = iftrue()
code = iftrue(TRUE, 1)
code = iftrue(2L, 1)
code = iftrue(FALSE, 1)
```

---

**jornada**

*Jornada rodent data*

**Description**

Counts of 17 rodent species across 24 sampling events, with the count being the total number observed across three trapping webs (146 traps in total) (Lightfoot *et al.* 2012).

**Usage**

```r
jornada
```

**Format**

A list of two `data.frame` objects with rows corresponding to documents (sampling events). One element is the document term table (called `document_term_table`), which contains counts of the species (terms) in each sample (document), and the other is the document covariate table (called `document_covariate_table`) with columns of covariates (time step, year, season).
Source

https://jornada.nmsu.edu/lter/dataset/49798/view

References


**LDATS**

*Package to conduct two-stage analyses combining Latent Dirichlet Allocation with Bayesian Time Series models*

Description


Documentation

- Technical mathematical manuscript
- End-user-focused vignette worked example
- Computational pipeline vignette
- Comparison to Christensen *et al.*

References


LDA_msg

Create the model-running-message for an LDA

Description

Produce and print the message for a given LDA model.

Usage

LDA_msg(mod_topics, mod_seeds, control = list())

Arguments

- mod_topics: integer value corresponding to the number of topics in the model.
- mod_seeds: integer value corresponding to the seed used for the model.
- control: Class LDA_controls list of control parameters to be used in LDA (note that "seed" will be overwritten).

Examples

LDA_msg(mod_topics = 4, mod_seeds = 2)

LDA_set

Run a set of Latent Dirichlet Allocation models

Description

For a given dataset consisting of counts of words across multiple documents in a corpus, conduct multiple Latent Dirichlet Allocation (LDA) models (using the Variational Expectation Maximization (VEM) algorithm; Blei et al. 2003) to account for [1] uncertainty in the number of latent topics and [2] the impact of initial values in the estimation procedure.

LDA_set is a list wrapper of LDA in the topicmodels package (Grun and Hornik 2011).

check_LDA_set_inputs checks that all of the inputs are proper for LDA_set (that the table of observations is conformable to a matrix of integers, the number of topics is an integer, the number of seeds is an integer and the controls list is proper).

Usage

LDA_set(document_term_table, topics = 2, nseeds = 1, control = list())

check_LDA_set_inputs(document_term_table, topics, nseeds, control)
LDA_set_control

Create control list for set of LDA models

Description

This function provides a simple creation and definition of the list used to control the set of LDA models. It is set up to be easy to work with the existing control capacity of LDA.

Usage

LDA_set_control(quiet = FALSE, measurer = AIC, selector = min, iseed = 2, ...)

Arguments

document_term_table
Table of observation count data (rows: documents, columns: terms. May be a class matrix or data.frame but must be conformable to a matrix of integers, as verified by check_document_term_table.

topics
Vector of the number of topics to evaluate for each model. Must be conformable to integer values.
nseeds
Number of seeds (replicate starts) to use for each value of topics. Must be conformable to integer value.
control
A list of parameters to control the running and selecting of LDA models. Values not input assume default values set by LDA_set_control. Values for running the LDAs replace defaults in (LDAcontrol, see LDA (but if seed is given, it will be overwritten; use iseed instead).

Value

LDA_set: list (class: LDA_set) of LDA models (class: LDA_VEM). check_LDA_set_inputs: an error message is thrown if any input is improper, otherwise NULL.

References


Examples

data(rodents)
lda_data <- rodents$document_term_table
r_LDA <- LDA_set(lda_data, topics = 2, nseeds = 2)
Arguments

quiet logical indicator of whether the model should run quietly.
measurer, selector Function names for use in evaluation of the LDA models. measurer is used to create a value for each model and selector operates on the values to choose the model(s) to pass on.
iseed integer initial seed for the model set.
... Additional arguments to be passed to LDA as a control input.

Value

list for controlling the LDA model fit.

Examples

LDA_set_control()

LDA_TS

Run a full set of Latent Dirichlet Allocations and Time Series models

Description

Conduct a complete LDATS analysis (Christensen et al. 2018), including running a suite of Latent Dirichlet Allocation (LDA) models (Blei et al. 2003, Grun and Hornik 2011) via LDA_set, selecting LDA model(s) via select_LDA, running a complete set of Bayesian Time Series (TS) models (Western and Kleykamp 2004) via TS_on_LDA on the chosen LDA model(s), and selecting the best TS model via select_TS.

conform_LDA_TS_data converts the data input to match internal and sub-function specifications.

check_LDA_TS_inputs checks that the inputs to LDA_TS are of proper classes for a full analysis.

Usage

LDA_TS(data, topics = 2, nseeds = 1, formulas = ~1, nchangepoints = 0, timename = "time", weights = TRUE, control = list())

conform_LDA_TS_data(data, quiet = FALSE)

check_LDA_TS_inputs(data = NULL, topics = 2, nseeds = 1, formulas = ~1, nchangepoints = 0, timename = "time", weights = TRUE, control = list())
Arguments

data Either a document term table or a list including at least a document term table (with the word "term" in the name of the element) and optionally also a document covariate table (with the word "covariate" in the name of the element).

The document term table is a table of observation count data (rows: documents, columns: terms) that may be a matrix or data.frame, but must be conformable to a matrix of integers, as verified by check_document_term_table.

The document covariate table is a table of associated data (rows: documents, columns: time index and covariate options) that may be a matrix or data.frame, but must be a conformable to a data table, as verified by check_document_covariate_table. Every model needs a covariate to describe the time value for each document (in whatever units and whose name in the table is input in timename) that dictates the application of the change points. If a covariate table is not provided, the model assumes the observations were equi-spaced in time. All covariates named within specific models in formulas must be included.

topics Vector of the number of topics to evaluate for each model. Must be conformable to integer values.

nseeds integer number of seeds (replicate starts) to use for each value of topics in the LDAs. Must be conformable to integer value.

formulas Vector of formula(s) for the continuous (non-change point) component of the time series models. Any predictor variable included in a formula must also be a column in the document_covariate_table. Each element (formula) in the vector is evaluated for each number of change points and each LDA model.

nchangepoints Vector of integer s corresponding to the number of change points to include in the time series models. 0 is a valid input corresponding to no change points (i.e., a singular time series model), and the current implementation can reasonably include up to 6 change points. Each element in the vector is the number of change points used to segment the data for each formula (entry in formulas) component of the TS model, for each selected LDA model.

timename character element indicating the time variable used in the time series. Defaults to "time". The variable must be integer-conformable or a Date. If the variable named is a Date, the input is converted to an integer, resulting in the timestep being 1 day, which is often not desired behavior.

weights Optional input for overriding standard weighting for documents in the time series. Defaults to TRUE, translating to an appropriate weighting of the documents based on the size (number of words) each document (the result of LDA is a matrix of proportions, which does not account for size differences among documents. Alternatively can be NULL for an equal weighting among documents or a numeric vector.

control A list of parameters to control the running and selecting of LDA and TS models. Values not input assume default values set by LDA_TS_control.

quiet logical indicator for conform_LDA_TS_data to indicate if messages should be printed.
LDA_TS_control

Create the controls list for the LDATS model

Description
Create and define a list of control options used to run the LDATS model, as implemented by LDA_TS.

Usage
LDA_TS_control(quiet = FALSE, measurer_LDA = AIC, selector_LDA = min, iseed = 2, memoise = TRUE, response = "gamma", lambda = 0, measurer_TS = AIC, selector_TS = min, ntemps = 6, penultimate_temp = 2^6, ultimate_temp = 1e+10, q = 0, nit = 10000, magnitude = 12, burnin = 0, thin_frac = 1, summary_prob = 0.95, seed = NULL, ...)

Value
LDA_TS: a class LDA_TS list object including all fitted LDA and TS models and selected models specifically as elements "LDA models" (from LDA_set), "Selected LDA model" (from select_LDA), "TS models" (from TS_on_LDA), and "Selected TS model" (from select_TS).

conform_LDA_TS_data: a data list that is ready for analyses using the stage-specific functions.

check_LDA_TS_inputs: an error message is thrown if any input is improper, otherwise NULL.

References


Examples
data(rodents)

mod <- LDA_TS(data = rodents, topics = 2, nseeds = 1, formulas = ~1, nchangepoints = 1, timename = "newmoon")

conform_LDA_TS_data(rodents)
check_LDA_TS_inputs(rodents, timename = "newmoon")
Arguments

quiet logical indicator of whether the model should run quietly.
measurer_LDA, selector_LDA
Function names for use in evaluation of the LDA models. measurer_LDA is used to create a value for each model and selector_LDA operates on the values to choose the model.
iseed integer initial seed for the LDA model set.
memoise logical indicator of whether the multinomial functions should be memoised (via memoise). Memoisation happens to both multinom_TS and multinom_TS_chunk.
response character element indicating the response variable used in the time series. Should be set to "gamma" for LDATS.
lambda numeric "weight" decay term used to set the prior on the regressors within each chunk-level model. Defaults to 0, corresponding to a fully vague prior.
measurer_TS, selector_TS
Function names for use in evaluation of the TS models. measurer_TS is used to create a value for each model and selector_TS operates on the values to choose the model.
ntemps integer number of temperatures (chains) to use in the ptMCMC algorithm.
penultimate_temp Penultimate temperature in the ptMCMC sequence.
ultimate_temp Ultimate temperature in the ptMCMC sequence.
q Exponent controlling the ptMCMC temperature sequence from the focal chain (reference with temperature = 1) to the penultimate chain. 0 (default) implies a geometric sequence. 1 implies squaring before exponentiating.
nit integer number of iterations (steps) used in the ptMCMC algorithm.
magnitude Average magnitude (defining a geometric distribution) for the proposed step size in the ptMCMC algorithm.
burnin integer number of iterations to remove from the beginning of the ptMCMC algorithm.
thin_frac Fraction of iterations to retain, from the ptMCMC. Must be (0, 1], and the default value of 1 represents no thinning.
summary_prob Probability used for summarizing the posterior distributions (via the highest posterior density interval, see HPDinterval) of the TS model.
seed Input to set.seed in the time series model for replication purposes.
... Additional arguments to be passed to LDA as a control input.

Value

list of control lists, with named elements LDAcontrol, TScontrol, and quiet.

Examples

LDA_TS_control()
**Description**

Imported but updated calculations from topicmodels package, as applied to Latent Dirichlet Allocation fit with Variational Expectation Maximization via \texttt{LDA}.

**Usage**

```r
## S3 method for class 'LDA_VEM'
logLik(object, ...) # S3 method for class 'LDA_VEM'
logLik(object, ...)
```

**Arguments**

- \texttt{object} A \texttt{LDA_VEM}-class object.
- \ldots Not used, simply included to maintain method compatibility.

**Details**

The number of degrees of freedom is 1 (for alpha) plus the number of entries in the document-topic matrix. The number of observations is the number of entries in the document-term matrix.

**Value**

Log likelihood of the model \texttt{logLik}, also with \texttt{df} (degrees of freedom) and \texttt{nobs} (number of observations) values.

**References**


**Examples**

```r
data(rodents)
lda_data <- rodents$document_term_table
r_LDA <- LDA_set(lda_data, topics = 2)
logLik(r_LDA[[1]])
```
Description

Convenience function to simply extract the logLik element (and df and nobs) from a multinom_TS_fit object fit by multinom_TS. Extends logLik from multinom to multinom_TS_fit objects.

Usage

```r
## S3 method for class 'multinom_TS_fit'
logLik(object, ...)
```

Arguments

- `object`: A multinom_TS_fit-class object.
- `...`: Not used, simply included to maintain method compatibility.

Value

Log likelihood of the model, as class logLik, with attributes df (degrees of freedom) and nobs (the number of weighted observations, accounting for size differences among documents).

Examples

```r
data(rodents)
dtt <- rodents$document_term_table
lda <- LDA_set(dtt, 2, 1, list(quiet = TRUE))
dct <- rodents$document_covariate_table
dct$gamma <- lda[[1]]@gamma
weights <- document_weights(dtt)
mts <- multinom_TS(dct, formula = gamma ~ 1, changepoints = c(20, 50),
                   timename = "newmoon", weights = weights)
logLik(mts)
```

Description

Convenience function to extract and format the log likelihood of a TS_fit-class object fit by multinom_TS.
Usage

```r
## S3 method for class 'TS_fit'
logLik(object, ...)
```

Arguments

- `object`: Class TS_fit object to be evaluated.
- `...`: Not used, simply included to maintain method compatibility.

Value

Log likelihood of the model logLik, also with df (degrees of freedom) and nobs (number of observations) values.

Examples

```r
data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)
TSmod <- TS(data, gamma ~ 1, nchangepoints = 1, "newmoon", weights)
logLik(TSmod)
```

---

**logsumexp**

*Calculate the log-sum-exponential (LSE) of a vector*

Description

Calculate the exponent of a vector (offset by the max), sum the elements, calculate the log, remove the offset.

Usage

```r
logsumexp(x)
```

Arguments

- `x`: numeric vector

Value

The LSE.
Examples

    logsumexp(1:10)

---

**memoise_fun**  
*Logical control on whether or not to memoise*

---

**Description**

This function provides a simple, logical toggle control on whether the function `fun` should be memoised via `memoise` or not.

**Usage**

    memoise_fun(fun, memoise_tf = TRUE)

**Arguments**

- `fun` Function name to (potentially) be memoised.
- `memoise_tf` logical value indicating if `fun` should be memoised.

**Value**

`fun`, memoised if desired.

**Examples**

    sum_memo <- memoise_fun(sum)

---

**messageq**  
*Optionally generate a message based on a logical input*

---

**Description**

Given the input to `quiet`, generate the message(s) in `msg` or not.

**Usage**

    messageq(msg = NULL, quiet = FALSE)

**Arguments**

- `msg` character vector of the message(s) to generate or NULL. If more than one element is contained in `msg`, they are concatenated with a newline between.
- `quiet` logical indicator controlling if the message is generated.
Examples
messageq("hello")
messageq("hello", TRUE)

mirror_vcov
Create a properly symmetric variance covariance matrix

Description
A wrapper on \texttt{vcov} to produce a symmetric matrix. If the default matrix returned by \texttt{vcov} is symmetric it is returned simply. If it is not, in fact, symmetric (as occurs occasionally with \texttt{multinom} applied to proportions), the matrix is made symmetric by averaging the lower and upper triangles. If the relative difference between the upper and lower triangles for any entry is more than 0.1

Usage
\texttt{mirror_vcov(x)}

Arguments
\texttt{x} Model object that has a defined method for \texttt{vcov}.

Value
Properly symmetric variance covariance matrix.

Examples
\begin{verbatim}
dat <- data.frame(y = rnorm(50), x = rnorm(50))
mod <- lm(dat)
mirror_vcov(mod)
\end{verbatim}

modalvalue
Determine the mode of a distribution

Description
Find the most common entry in a vector. Ties are not allowed, the first value encountered within the modal set if there are ties is deemed the mode.

Usage
\texttt{modalvalue(x)}
Arguments

x numeric vector.

Value

Numeric value of the mode.

Examples

d1 <- c(1, 1, 1, 2, 2, 3)
modalvalue(d1)

multinom_TS

Fit a multinomial change point Time Series model

Description

Fit a set of multinomial regression models (via `multinom`, Venables and Ripley 2002) to a time series of data divided into multiple segments (a.k.a. chunks) based on given locations for a set of change points.

check_multinom_TS_inputs checks that the inputs to multinom_TS are of proper classes for an analysis.

Usage

multinom_TS(data, formula, changepoints = NULL, timename = "time",
weights = NULL, control = list())

check_multinom_TS_inputs(data, formula = gamma ~ 1,
changepoints = NULL, timename = "time", weights = NULL,
control = list())

Arguments

data data.frame including [1] the time variable (indicated in timename), [2] the predictor variables (required by formula) and [3], the multinomial response variable (indicated in formula) as verified by `check_timename` and `check_formula`. Note that the response variables should be formatted as a data.frame object named as indicated by the response entry in the control list, such as gamma for a standard TS analysis on LDA output. See Examples.

formula formula defining the regression between relationship the change points. Any predictor variable included must also be a column in data and any (multinomial) response variable must be a set of columns in data, as verified by `check_formula`. 
**multinom_TS**

- **changepoints**: Numeric vector indicating locations of the change points. Must be conformable to integer values. Validity checked by `check_changepoints` and `verify_changepoint_locations`.

- **timename**: character element indicating the time variable used in the time series. Defaults to "time". The variable must be integer-conformable or a Date. If the variable named is a Date, the input is converted to an integer, resulting in the timestep being 1 day, which is often not desired behavior.

- **weights**: Optional class numeric vector of weights for each document. Defaults to NULL, translating to an equal weight for each document. When using multinom_TS in a standard LDATS analysis, it is advisable to weight the documents by their total size, as the result of LDA is a matrix of proportions, which does not account for size differences among documents. For most models, a scaling of the weights (so that the average is 1) is most appropriate, and this is accomplished using `document_weights`.

- **control**: A list of parameters to control the fitting of the Time Series model including the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values not input assume defaults set by `TS_control`.

**Value**

- `multinom_TS`: Object of class `multinom_TS_fit`, which is a list of [1] chunk-level model fits ("chunk models"), [2] the total log likelihood combined across all chunks ("logLik"), and [3] a `data.frame` of chunk beginning and ending times ("logLik" with columns "start" and "end").

- `check_multinom_TS_inputs`: an error message is thrown if any input is improper, otherwise NULL.

**References**


**Examples**

```r
data(rodents)
dtt <- rodents$document_term_table
lda <- LDA_set(dtt, 2, 1, list(quiet = TRUE))
dct <- rodents$document_covariate_table
dc$gamma <- lda[[1]]$gamma
weights <- document_weights(dtt)
check_multinom_TS_inputs(dct, timename = "newmoon")
mts <- multinom_TS(dct, formula = gamma ~ 1, changepoints = c(20,50),
                   timename = "newmoon", weights = weights)
```
**multinom_TS_chunk**

*Fit a multinomial Time Series model chunk*

**Description**

Fit a multinomial regression model (via `multinom`, Ripley 1996, Venables and Ripley 2002) to a defined chunk of time (a.k.a. segment) \([\text{chunk}^{\text{start}}, \text{chunk}^{\text{end}}]\) within a time series.

**Usage**

```
multinom_TS_chunk(data, formula, chunk, timename = "time", weights = NULL, control = list())
```

**Arguments**

- **data**
  - Class `data.frame` object including the predictor and response variables.

- **formula**
  - Formula as a `formula` or `character` object describing the chunk.

- **chunk**
  - Length-2 vector of times: \([1\text{]} \text{start} \text{,} 2\text{]} \text{end}\), the start time for the chunk and the end time for the chunk.

- **timename**
  - Character element indicating the time variable used in the time series. Defaults to "time". The variable must be integer-conformable or a `Date`. If the variable named is a `Date`, the input is converted to an integer, resulting in the timestep being 1 day, which is often not desired behavior.

- **weights**
  - Optional class `numeric` vector of weights for each document. Defaults to `NULL`, translating to an equal weight for each document. When using `multinom_TS` in a standard LDATS analysis, it is advisable to weight the documents by their total size, as the result of `LDA` is a matrix of proportions, which does not account for size differences among documents. For most models, a scaling of the weights (so that the average is 1) is most appropriate, and this is accomplished using `document_weights`.

- **control**
  - A list of parameters to control the fitting of the Time Series model including the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values not input assume defaults set by `TS_control`.

**Value**

Fitted model object for the chunk, of classes `multinom` and `nnet`.

**References**


normalize

Normalize a vector

Description

Normalize a numeric vector to be on the scale of [0,1].

Usage

normalize(x)

Arguments

x numeric vector.

Value

Normalized x.

Examples

normalize(1:10)

package_chunk_fits

Package the output of the chunk-level multinomial models into a multinom_TS_fit list

Description

Takes the list of fitted chunk-level models returned from TS_chunk_memo (the memoised version of multinom_TS_chunk) and packages it as a multinom_TS_fit object. This involves naming the model fits based on the chunk time windows, combining the log likelihood values across the chunks, and setting the class of the output object.
Usage

package_chunk_fits(chunks, fits)

Arguments

- **chunks**: Data frame of start and end times for each chunk (row).
- **fits**: List of chunk-level fits returned by TS_chunk_memo, the memoised version of multinom_TS_chunk.

Value

Object of class multinom_TS_fit, which is a list of [1] chunk-level model fits, [2] the total log likelihood combined across all chunks, and [3] the chunk time data table.

Examples

```r
data(rodents)
dtt <- rodents$document_term_table
lda <- LDA_set(dtt, 2, 1, list(quiet = TRUE))
dct <- rodents$document_covariate_table
dct$gamma <- lda[[1]]@gamma
weights <- document_weights(dtt)
formula <- gamma ~ 1
changepoints <- c(20,50)
timename <- "newmoon"
TS_chunk_memo <- memoise_fun(multinom_TS_chunk, TRUE)
chunks <- prep_chunks(dct, changepoints, timename)
nchunks <- nrow(chunks)
fits <- vector("list", length = nchunks)
for (i in 1:nchunks){
  fits[[i]] <- TS_chunk_memo(dct, formula, chunks[i, ], timename,
                           weights, TS_control())
}
package_chunk_fits(chunks, fits)
```

---

**package_LDA_set**

**Package the output from LDA_set**

**Description**

Name the elements (LDA models) and set the class (LDA_set) of the models returned by LDA_set.

**Usage**

```r
package_LDA_set(mods, mod_topics, mod_seeds)
```
Arguments

- mods: Fitted models returned from LDA.
- mod_topics: Vector of integer values corresponding to the number of topics in each model.
- mod_seeds: Vector of integer values corresponding to the seed used for each model.

Value

lis (class: LDA_set) of LDA models (class: LDA_VEM).

Examples

data(rodents)
document_term_table <- rodents$document_term_table
topics <- 2
nseeds <- 2
control <- LDA_set_control()
mod_topics <- rep(topics, each = length(seq(2, nseeds * 2, 2)))
iseed <- control$iseed
mod_seeds <- rep(seq(iseed, iseed + (nseeds - 1)* 2, 2), length(topics))
nmods <- length(mod_topics)
mods <- vector("list", length = nmods)
for (i in 1:nmods){
    LDA_msg(mod_topics[i], mod_seeds[i], control)
    control_i <- prep_LDA_control(seed = mod_seeds[i], control = control)
    mods[[i]] <- topicmodels::LDA(document_term_table, k = mod_topics[i],
                                 control = control_i)
}
package_LDA_set(mods, mod_topics, mod_seeds)
Arguments

LDAs  List (class: LDA_set) of LDA models (class: LDA), as returned by LDA_set.

sel_LDA  A reduced version of LDAs that only includes the LDA model(s) selected by select_LDA. Still should be of class LDA_set.

TSs  Class TS_on_LDA list of results from TS applied for each model on each LDA model input, as returned by TS_on_LDA.

sel_TSS  A reduced version of TSs (of class TS_fit) that only includes the TS model chosen via select_TS.

Value

Class LDA_TS-class object including all fitted models and selected models specifically, ready to be returned from LDA_TS.

Examples

data(rodents)
data <- rodents
control <- LDA_TS_control()
dtt <- data$document_term_table
dct <- data$document_covariate_table
weights <- document_weights(dtt)
LDAs <- LDA_set(dtt, 2, 1, control$LDA_set_control)
sel_LDA <- select_LDA(LDAs, control$LDA_set_control)
TSs <- TS_on_LDA(sel_LDA, dct, ~1, 1, "newmoon", weights, control$TS_control)
sel_TSS <- select_TS(TSS, control$TS_control)
package_LDA_TS(LDAs, sel_LDA, TSS, sel_TSS)

package_TS  Summarize the Time Series model

Description

Calculate relevant summaries for the run of a Time Series model within TS and package the output as a TS_fit-class object.

Usage

package_TS(data, formula, timename, weights, control, rho_dist, eta_dist)
Arguments

data data.frame including [1] the time variable (indicated in timename), [2] the predictor variables (required by formula) and [3], the multinomial response variable (indicated in formula) as verified by check_timename and check_formula. Note that the response variables should be formatted as a data.frame object named as indicated by the response entry in the control list, such as gamma for a standard TS analysis on LDA output.

formula formula defining the regression between relationship the change points. Any predictor variable included must also be a column in data and any (multinomial) response variable must be a set of columns in data, as verified by check_formula.

timename character element indicating the time variable used in the time series.

weights Optional class numeric vector of weights for each document. Defaults to NULL, translating to an equal weight for each document. When using multinom_TS in a standard LDATS analysis, it is advisable to weight the documents by their total size, as the result of LDA is a matrix of proportions, which does not account for size differences among documents. For most models, a scaling of the weights (so that the average is 1) is most appropriate, and this is accomplished using document_weights.

control A list of parameters to control the fitting of the Time Series model including the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values not input assume defaults set by TS_control.

rho_dist List of saved data objects from the ptMCMC estimation of change point locations returned by est_changepoints (unless nchangepoints is 0, then NULL).

eta_dist Matrix of draws (rows) from the marginal posteriors of the coefficients across the segments (columns), as estimated by est_regressors.

Value

TS_fit-class list containing the following elements, many of which are hidden for printing, but are accessible:

data data input to the function.

formula formula input to the function.

nchangepoints nchangepoints input to the function.

weights weights input to the function.

timename timename input to the function.

control control input to the function.

lls Iteration-by-iteration logLik values for the full time series fit by multinom_TS.

rhos Iteration-by-iteration change point estimates from est_changepoints.

etas Iteration-by-iteration marginal regressor estimates from est_regressors, which have been unconditioned with respect to the change point locations.

ptMCMC_diagnostics ptMCMC diagnostics, see diagnose_ptMCMC

rho_summary Summary table describing rhos (the change point locations), see summarize_rhos.
rho_vcov  Variance-covariance matrix for the estimates of rhos (the change point locations), see measure_rho_vcov.

eta_summary  Summary table describing ets (the regressors), see summarize_etas.

eta_vcov  Variance-covariance matrix for the estimates of etas (the regressors), see measure_eta_vcov.

logLik  Across-iteration average of log-likelihoods (lls).

nparams  Total number of parameters in the full model, including the change point locations and regressors.

AIC  Penalized negative log-likelihood, based on logLik and nparams.

Examples

data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)
formula <- gamma ~ 1
nchangepoints <- 1
control <- TS_control()
data <- data[order(data[, "newmoon"], ),]
rho_dist <- est_changepoints(data, formula, nchangepoints, "newmoon", weights, control)
eta_dist <- est_regressors(rho_dist, data, formula, "newmoon", weights, control)
package_TS(data, formula, "newmoon", weights, control, rho_dist, eta_dist)
Arguments

- **TSmods**: list of results from TS applied for each model on each LDA model input.
- **LDA_models**: List of LDA models (class LDA_set, produced by LDA_set) or a singular LDA model (class LDA, produced by LDA).
- **models**: data.frame object returned from expand_TS that contains the combinations of LDA models, and formulas and nchangepoints used in the TS models.

Value

Class TS_on_LDA list of results from TS applied for each model on each LDA model input.

Examples

```r
data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDAs <- LDA_set(document_term_table, topics = 2:3, nseeds = 2)
LDA_models <- select_LDA(LDAs)
weights <- document_weights(document_term_table)
mods <- expand_TS(LDA_models, c(~ 1, ~ newmoon), 0:1)
nmods <- nrow(mods)
TSmods <- vector("list", nmods)
for(i in 1:nmods){
  formula_i <- mods$formula[[i]]
  nchangepoints_i <- mods$nchangepoints[i]
  data_i <- prep_TS_data(document_covariate_table, LDA_models, mods, i)
  TSmods[[i]] <- TS(data_i, formula_i, nchangepoints_i, "newmoon", weights, TS_control())
}
package_TS_on_LDA(TSmods, LDA_models, mods)
```

---

**plot.LDA_set**

Plot a set of LDATS LDA models

Description

Generalization of the plot function to work on a list of LDA topic models (class LDA_set).

Usage

```r
## S3 method for class 'LDA_set'
plot(x, ...)
```
Arguments

x  An LDA_set object of LDA topic models.
...

Additional arguments to be passed to subfunctions.

Value

NULL.

Examples

```r
data(rodents)
lda_data <- rodents$document_term_table
r_LDA <- LDA_set(lda_data, topics = 2, nseeds = 2)
plot(r_LDA)
```

plot.LDA_TS  

Plot the results from a full LDATS analysis

Description

Generalization of the `plot` function to work on fitted LDA_TS model objects (class LDA_TS) returned by LDA_TS.

Usage

```r
## S3 method for class 'LDA_TS'
plot(x, ..., cols = set_LDA_TS_plot_cols(),
     bin_width = 1, xname = NULL, border = NA, selection = "median")
```

Arguments

x  A LDA_TS object of a full LDATS model fit by LDA_TS.
...

Additional arguments to be passed to subfunctions. Not currently used, just retained for alignment with plot.

cols  list of elements used to define the colors for the two panels of the summary plot, as generated simply using `set_LDA_TS_plot_cols`. cols has two elements: LDA and TS, each corresponding the set of plots for its stage in the full model. LDA contains entries cols and option (see `set_LDA_plot_colors`). TS contains two entries, rho and gamma, each corresponding to the related panel, and each containing default values for entries named cols, option, and alpha (see `set_TS_summary_plot_cols`, `set_gamma_colors`, and `set_rho_hist_colors`).

bin_width  Width of the bins used in the histograms of the summary time series plot, in units of the time variable used to fit the model (the x-axis).
**plot.LDA_VEM**

*xname*  Label for the x-axis in the summary time series plot. Defaults to NULL, which results in usage of the timename element of the control list (held in control$TS_control$timename). To have no label printed, set xname = "".

*border*  Border for the histogram, default is NA.

*selection*  Indicator of the change points to use in the time series summary plot. Currently only defined for "median" and "mode".

**Value**

NULL.

**Examples**

```r
data(rodents)
mod <- LDA_TS(data = rodents, topics = 2, nseeds = 1, formulas = ~1,
             nchangepoints = 1, timename = "newmoon")
plot(mod, binwidth = 5, xlab = "New moon")
```

---

**plot.LDA_VEM**  
*Plot the results of an LDATS LDA model*

**Description**

Create an LDATS LDA summary plot, with a top panel showing the topic proportions for each word and a bottom panel showing the topic proportions of each document/over time. The plot function is defined for class LDA_VEM specifically (see LDA).

**LDA_plot_top_panel** creates an LDATS LDA summary plot top panel showing the topic proportions word-by-word.

**LDA_plot_bottom_panel** creates an LDATS LDA summary plot bottom panel showing the topic proportions over time/documents.

**Usage**

```r
## S3 method for class 'LDA_VEM'
plot(x, ..., xtime = NULL, xname = NULL,
     cols = NULL, option = "C", alpha = 0.8, LDATS = FALSE)

LDA_plot_top_panel(x, cols = NULL, option = "C", alpha = 0.8,
                   together = FALSE, LDATS = FALSE)

LDA_plot_bottom_panel(x, xtime = NULL, xname = NULL, cols = NULL,
                       option = "C", alpha = 0.8, together = FALSE, LDATS = FALSE)
```
Arguments

x
Object of class LDA_VEM.

... Not used, retained for alignment with base function.

xtime
Optional x values used to plot the topic proportions according to a specific time value (rather than simply the order of observations).

xname
Optional name for the x values used in plotting the topic proportions (otherwise defaults to "Document").

cols
Colors to be used to plot the topics. Any valid color values (e.g., see colors, rgb) can be input as with a standard plot. The default (cols = NULL) triggers use of viridis color options (see option).

option
A character string indicating the color option from viridis to use if 'cols == NULL': Four options are available: "magma" (or "A"), "inferno" (or "B"), "plasma" (or "C", the default option), "viridis" (or "D") and "cividis" (or "E").

alpha
Numeric value [0,1] that indicates the transparency of the colors used. Supported only on some devices, see rgb.

LDATS logical indicating if the LDA plot is part of a larger LDATS plot output.

together logical indicating if the subplots are part of a larger LDA plot output.

Value

NULL.

Examples

data(rodents)
lda_data <- rodents$document_term_table
r_LDA <- LDA_set(lda_data, topics = 4, nseeds = 10)
best_lda <- select_LDA(r_LDA)[[1]]
plot(best_lda, option = "cividis")
LDA_plot_top_panel(best_lda, option = "cividis")
LDA_plot_bottom_panel(best_lda, option = "cividis")

plot.TS_fit

Plot an LDATS TS model

Description

Generalization of the plot function to work on fitted TS model objects (class TS_fit) returned from TS.
Usage

## S3 method for class 'TS_fit'
plot(x, ..., plot_type = "summary",
     interactive = FALSE, cols = set_TS_summary_plot_cols(),
     bin_width = 1, xname = NULL, border = NA, selection = "median",
     LDATS = FALSE)

Arguments

- **x**: A TS_fit object of a multinomial time series model fit by TS.
- **...**: Additional arguments to be passed to subfunctions. Not currently used, just retained for alignment with plot.
- **plot_type**: "diagnostic" or "summary".
- **interactive**: logical input, should be codeTRUE unless testing.
- **cols**: list of elements used to define the colors for the two panels of the summary plot, as generated simply using set_TS_summary_plot_cols. cols has two elements rho and gamma, each corresponding to the related panel, and each containing default values for entries named cols, option, and alpha. See set_gamma_colors and set_rho_hist_colors for details on usage.
- **bin_width**: Width of the bins used in the histograms of the summary time series plot, in units of the x-axis (the time variable used to fit the model).
- **xname**: Label for the x-axis in the summary time series plot. Defaults to NULL, which results in usage of the timename element of the control list (held incontrol$TS_control$timename). To have no label printed, set xname = "".
- **border**: Border for the histogram, default is NA.
- **selection**: Indicator of the change points to use in the time series summary plot. Currently only defined for "median" and "mode".
- **LDATS**: logical indicating if the plot is part of a larger LDATS plot output.

Value

NULL.

Examples

data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)
TSmod <- TS(data, gamma ~ 1, nchangepoints = 1, "newmoon", weights)
plot(TSmod)
posterior_plot

Produce the posterior distribution histogram panel for the TS diagnostic plot of a parameter

Description

Produce a vanilla histogram plot using hist for the parameter of interest (rho or eta) as part of TS_diagnostics_plot. A vertical line is added to show the median of the posterior.

Usage

posterior_plot(x, xlab = "parameter value")

Arguments

x

Vector of parameter values drawn from the posterior distribution, indexed to the iteration by the order of the vector.

xlab

character value used to label the x axis.

Value

NULL.

Examples

posterior_plot(rnorm(100, 0, 1))

prep_chunks

Prepare the time chunk table for a multinomial change point Time Series model

Description

Creates the table containing the start and end times for each chunk within a time series, based on the change points (used to break up the time series) and the range of the time series. If there are no change points (i.e. changepoints is NULL, there is still a single chunk defined by the start and end of the time series.

Usage

prep_chunks(data, changepoints = NULL, timename = "time")
**prep_cpts**  

**Arguments**

- **data**: Class `data.frame` object including the predictor and response variables, but specifically here containing the column indicated by the `timename` input.

- **changepoints**: Numeric vector indicating locations of the change points. Must be conformable to integer values.

- **timename**: Character element indicating the time variable used in the time series. Defaults to "time". The variable must be integer-conformable or a Date. If the variable named is a Date, the input is converted to an integer, resulting in the timestep being 1 day, which is often not desired behavior.

**Value**

A `data.frame` of start and end times (columns) for each chunk (rows).

**Examples**

```r
data(rodents)
dtt <- rodents$document_term_table
lda <- LDA_set(dtt, 2, 1, list(quiet = TRUE))
dct <- rodents$document_covariate_table
dct$gamma <- lda[[1]]$gamma
chunks <- prep_chunks(dct, changepoints = 100, timename = "newmoon")
```

---

**prep_cpts** *Initialize and update the change point matrix used in the ptMCMC algorithm*

**Description**

Each of the chains is initialized by `prep_cpts` using a draw from the available times (i.e. assuming a uniform prior), the best fit (by likelihood) draw is put in the focal chain with each subsequently worse fit placed into the subsequently hotter chain. `update_cpts` updates the change points after every iteration in the ptMCMC algorithm.

**Usage**

```r
prep_cpts(data, formula, nchangepoints, timename, weights, 
control = list())

update_cpts(cpts, swaps)
```
Arguments

- **data**: data.frame including [1] the time variable (indicated in timename), [2] the predictor variables (required by formula) and [3], the multinomial response variable (indicated in formula) as verified by check_timename and check_formula. Note that the response variables should be formatted as a data.frame object named as indicated by the response entry in the control list, such as gamma for a standard TS analysis on LDA output.

- **formula**: formula defining the regression relationship between the change points, see formula. Any predictor variable included must also be a column in data and any (multinomial) response variable must be a set of columns in data, as verified by check_formula.

- **nchangepoints**: integer corresponding to the number of change points to include in the model. 0 is a valid input (corresponding to no change points, so a singular time series model), and the current implementation can reasonably include up to 6 change points. The number of change points is used to dictate the segmentation of the data for each continuous model and each LDA model.

- **timename**: character element indicating the time variable used in the time series. Defaults to "time". The variable must be integer-conformable or a Date. If the variable named is a Date, the input is converted to an integer, resulting in the timestep being 1 day, which is often not desired behavior.

- **weights**: Optional class numeric vector of weights for each document. Defaults to NULL, translating to an equal weight for each document. When using multinom_TS in a standard LDATS analysis, it is advisable to weight the documents by their total size, as the result of LDA is a matrix of proportions, which does not account for size differences among documents. For most models, a scaling of the weights (so that the average is 1) is most appropriate, and this is accomplished using document_weihts.

- **control**: A list of parameters to control the fitting of the Time Series model including the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values not input assume defaults set by TS_control.

- **cpts**: The existing matrix of change points.

- **swaps**: Chain configuration after among-temperature swaps.

Value

- list of [1] matrix of change points (rows) for each temperature (columns) and [2] vector of log-likelihood values for each of the chains.

Examples

```r
data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma```
prep_ids <- document_weights(document_term_table)
data <- data[order(data[, "newmoon"]), ]
saves <- prep_saves(1, TS_control())
inputs <- prep_ptMCMC_inputs(data, gamma ~ 1, 1, "newmoon", weights, TS_control())
cpts <- prep_cpts(data, gamma ~ 1, 1, "newmoon", weights, TS_control())
ids <- prep_ids(TS_control())
for(i in 1:TS_control()$nit){
  steps <- step_chains(i, cpts, inputs)
  swaps <- swap_chains(steps, inputs, ids)
  saves <- update_saves(i, saves, steps, swaps)
  cpts <- update_cpts(cpts, swaps)
  ids <- update_ids(ids, swaps)
}

prep_ids  Initialize and update the chain ids throughout the ptMCMC algorithm

Description

prep_ids creates and update_ids updates the active vector of identities (ids) for each of the chains in the ptMCMC algorithm. These ids are used to track trips of the particles among chains.

These functions were designed to work within TS and specifically est_changepoints, but have been generalized and would work within any general ptMCMC as long as control, ids, and swaps are formatted properly.

Usage

prep_ids(control = list())
update_ids(ids, swaps)

Arguments

control A list of parameters to control the fitting of the Time Series model including the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values not input assume defaults set by TS_control.
ids The existing vector of chain ids.
swaps Chain configuration after among-temperature swaps.

Value

The vector of chain ids.
Examples

```r
prep_ids()
data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)
data <- data[order(data[,"newmoon"], ), ]
saves <- prep_saves(1, TS_control())
inputs <- prep_ptMCMC_inputs(data, gamma ~ 1, 1, "newmoon", weights, TS_control())
cpts <- prep_cpts(data, gamma ~ 1, 1, "newmoon", weights, TS_control())
ids <- prep_ids(TS_control())
for(i in 1:TS_control()$nit){
  steps <- step_chains(i, cpts, inputs)
  swaps <- swap_chains(steps, inputs, ids)
  saves <- update_saves(i, saves, steps, swaps)
  cpts <- update_cpts(cpts, swaps)
  ids <- update_ids(ids, swaps)
}
```

---

**prep_LDA_control**

Set the control inputs to include the seed

Description

Update the control list for the LDA model with the specific seed as indicated. And remove controls not used within the LDA itself.

Usage

```r
prep_LDA_control(seed, control = list())
```

Arguments

- **seed** integer used to set the seed of the specific model.
- **control** Named list of control parameters to be used in LDA. Note that if control has an element named seed it will be overwritten by the seed argument of prep_LDA_control.

Value

list of controls to be used in the LDA.
**prep_pbar**

**Examples**

```r
prep_LDA_control(seed = 1)
```

---

**Description**

`prep_pbar` creates and `update_pbar` steps through the progress bars (if desired) in `TS`.

**Usage**

```r
prep_pbar(control = list(), bar_type = "rho", nr = NULL)
update_pbar(pbar, control = list())
```

**Arguments**

- `control` A list of parameters to control the fitting of the Time Series model including the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values not input assume defaults set by `TS_control`. Of use here is `quiet` which is a logical indicator of whether there should be information (i.e. the progress bar) printed during the run or not. Default is `TRUE`.
- `bar_type` "rho" (for change point locations) or "eta" (for regressors).
- `nr` integer number of unique realizations, needed when `bar_type = "eta"`.
- `pbar` The progress bar object returned from `prep_pbar`.

**Value**

- `prep_pbar`: the initialized progress bar object.
- `update_pbar`: the ticked-forward `pbar`.

**Examples**

```r
pb <- prep_pbar(control = list(nit = 2)); pb
pb <- update_pbar(pb); pb
pb <- update_pbar(pb); pb
```
prep_proposal_dist  

_pre-calculate the change point proposal distribution for the ptMCMC algorithm_

Description

Calculate the proposal distribution in advance of actually running the ptMCMC algorithm in order to decrease computation time. The proposal distribution is a joint of three distributions: [1] a multinomial distribution selecting among the change points within the chain, [2] a binomial distribution selecting the direction of the step of the change point (earlier or later in the time series), and [3] a geometric distribution selecting the magnitude of the step.

Usage

prep_proposal_dist(nchangepoints, control = list())

Arguments

nchangepoints  

Integer corresponding to the number of change points to include in the model. 0 is a valid input (corresponding to no change points, so a singular time series model), and the current implementation can reasonably include up to 6 change points. The number of change points is used to dictate the segmentation of the data for each continuous model and each LDA model.

control  

A list of parameters to control the fitting of the Time Series model including the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values not input assume defaults set by TS_control. Currently relevant here is magnitude, which controls the magnitude of the step size (is the average of the geometric distribution).

Value

list of two matrix elements: [1] the size of the proposed step for each iteration of each chain and [2] the identity of the change point location to be shifted by the step for each iteration of each chain.

Examples

prep_proposal_dist(nchangepoints = 2)
**prep_ptMCMC_inputs**

Prepare the inputs for the ptMCMC algorithm estimation of change points

**Description**

Package the static inputs (controls and data structures) used by the ptMCMC algorithm in the context of estimating change points.

This function was designed to work within TS and specifically `est_changepoints`. It is still hardcoded to do so, but has the capacity to be generalized to work with any estimation via ptMCMC with additional coding work.

**Usage**

```r
prep_ptMCMC_inputs(data, formula, nchangepoints, timename,
                   weights = NULL, control = list())
```

**Arguments**

- `data` Class `data.frame` object including [1] the time variable (indicated in `control`), [2] the predictor variables (required by `formula`) and [3], the multinomial response variable (indicated in `formula`).
- `formula` formula describing the continuous change. Any predictor variable included must also be a column in the data. Any (multinomial) response variable must also be a set of columns in `data`.
- `nchangepoints` Integer corresponding to the number of change points to include in the model. 0 is a valid input (corresponding to no change points, so a singular time series model), and the current implementation can reasonably include up to 6 change points. The number of change points is used to dictate the segmentation of the data for each continuous model and each LDA model.
- `timename` character element indicating the time variable used in the time series. Defaults to "time". The variable must be integer-conformable or a `Date`. If the variable named is a `Date`, the input is converted to an integer, resulting in the timestep being 1 day, which is often not desired behavior.
- `weights` Optional class numeric vector of weights for each document. Defaults to `NULL`, translating to an equal weight for each document. When using `multinom_TS` in a standard LDATS analysis, it is advisable to weight the documents by their total size, as the result of `LDA` is a matrix of proportions, which does not account for size differences among documents. For most models, a scaling of the weights (so that the average is 1) is most appropriate, and this is accomplished using `document_weights`.
- `control` A list of parameters to control the fitting of the Time Series model including the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values not input assume defaults set by `TS_control`. 
prep_saves

Value

Class ptMCMC_inputs list, containing the static inputs for use within the ptMCMC algorithm for estimating change points.

Examples

```r
data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)
data <- data[order(data[, "newmoon"]), ]
saves <- prep_saves(1, TS_control())
inputs <- prep_ptMCMC_inputs(data, gamma ~ 1, 1, "newmoon", weights,
TS_control())
```

Description

prep_saves creates the data structure used to save the output from each iteration of the ptMCMC algorithm, which is added via update_saves. Once the ptMCMC is complete, the saved data objects are then processed (burn-in iterations are dropped and the remaining iterations are thinned) via process_saves.

This set of functions was designed to work within TS and specifically est_changepoints. They are still hardcoded to do so, but have the capacity to be generalized to work with any estimation via ptMCMC with additional coding work.

Usage

```r
prep_saves(nchangepoints, control = list())
update_saves(i, saves, steps, swaps)
process_saves(saves, control = list())
```

Arguments

- `nchangepoints` integer corresponding to the number of change points to include in the model. 0 is a valid input (corresponding to no change points, so a singular time series model), and the current implementation can reasonably include up to 6 change points. The number of change points is used to dictate the segmentation of the data for each continuous model and each LDA model.
prep_temp_sequence

control A list of parameters to control the fitting of the Time Series model including the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values not input assume defaults set by TS_control.

i integer iteration index.
saves The existing list of saved data objects.
steps Chain configuration after within-temperature steps.
swaps Chain configuration after among-temperature swaps.

Value

list of ptMCMC objects: change points ($cpts$), log-likelihoods ($lls$), chain ids ($ids$), step acceptances ($step_accepts$), and swap acceptances ($swap_accepts$).

Examples

data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)
data <- data[order(data[, "newmoon"], ),]
saves <- prep_saves(1, TS_control())
inputs <- prep_ptMCMC_inputs(data, gamma ~ 1, 1, "newmoon", weights, TS_control())
cpts <- prep_cpts(data, gamma ~ 1, 1, "newmoon", weights, TS_control())
ids <- prep_ids(TS_control())
for(i in 1:TS_control()$nit){
    steps <- step_chains(i, cpts, inputs)
    swaps <- swap_chains(steps, inputs, ids)
    saves <- update_saves(i, saves, steps, swaps)
    cpts <- update_cpts(cpts, swaps)
    ids <- update_ids(ids, swaps)
}
process_saves(saves, TS_control())

Description

Create the series of temperatures used in the ptMCMC algorithm.

This function was designed to work within TS and est_changepoints specifically, but has been
prep_TS_data

generalized and would work with any ptMCMC model as long as control includes the relevant control parameters (and provided that the check_control function and its use here are generalized).

Usage

prep_temp_sequence(control = list())

Arguments

control A list of parameters to control the fitting of the Time Series model including the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values not input assume defaults set by TS_control.

Value

vector of temperatures.

Examples

prep_temp_sequence()

prep_TS_data

Prepare the model-specific data to be used in the TS analysis of LDA output

Description

Append the estimated topic proportions from a fitted LDA model to the document covariate table to create the data structure needed for TS.

Usage

prep_TS_data(document_covariate_table, LDA_models, mods, i = 1)

Arguments

document_covariate_table Document covariate table (rows: documents, columns: time index and covariate options). Every model needs a covariate to describe the time value for each document (in whatever units and whose name in the table is input in timename) that dictates the application of the change points. In addition, all covariates named within specific models in formula must be included. Must be a conformable to a data table, as verified by check_document_covariate_table.

LDA_models List of LDA models (class LDA_set, produced by LDA_set) or a singular LDA model (class LDA, produced by LDA).
mods

The data.table created by `expand_TS` that contains each of the models (defined by the LDA model to use and the and formula number of changepoints for the TS model). Indexed here by i.

i

integer index referencing the row in mods to use.

Value

Class `data.frame` object including [1] the time variable (indicated in `control`), [2] the predictor variables (required by `formula`) and [3], the multinomial response variable (indicated in `formula`), ready for input into TS.

Examples

```r
data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDAs <- LDA_set(document_term_table, topics = 2:3, nseeds = 2)
LDA_models <- select_LDA(LDAs)
weights <- document_weights(document_term_table)
formulas <- c(~ 1, ~ newmoon)
mods <- expand_TS(LDA_models, formulas = ~1, nchangepoints = 0)
data1 <- prep_TS_data(document_covariate_table, LDA_models, mods)
```

---

**print.LDA_TS**

*Print the selected LDA and TS models of LDA_TS object*

**Description**

Convenience function to print only the selected elements of a LDA_TS-class object returned by LDA_TS

**Usage**

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

**Arguments**

- `x` Class LDA_TS object to be printed.
- `...` Not used, simply included to maintain method compatibility.

**Value**

The selected models in x as a two-element list with the TS component only returning the non-hidden components.
Examples

```r
data(rodents)
mod <- LDA_TS(data = rodents, topics = 2, nseeds = 1, formulas = ~1,
nchangepoints = 1, timename = "newmoon")
print(mod)
```

---

### print.TS_fit

Print a Time Series model fit

#### Description

Convenience function to print only the most important components of a TS_fit-class object fit by TS.

#### Usage

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

#### Arguments

- `x`  
  Class TS_fit object to be printed.
- `...`  
  Not used, simply included to maintain method compatibility.

#### Value

The non-hidden parts of `x` as a list.

#### Examples

```r
data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)
TSMOD <- TS(data, gamma ~ 1, nchangepoints = 1, "newmoon", weights)
print(TSMOD)
```
print.TS_on_LDA

Print a set of Time Series models fit to LDAs

Description

Convenience function to print only the names of a TS_on_LDA-class object generated by TS_on_LDA.

Usage

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

Arguments

x Class TS_on_LDA object to be printed.
...
Not used, simply included to maintain method compatibility.

Value

character vector of the names of x’s models.

Examples

data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDAs <- LDA_set(document_term_table, topics = 2:3, nseeds = 2)
LDA_models <- select_LDA(LDAs)
weights <- document_weights(document_term_table)
formulas <- c(~ 1, ~ newmoon)
mods <- TS_on_LDA(LDA_models, document_covariate_table, formulas,
              nchangepoints = 0:1, timename = "newmoon", weights)
print(mods)

print_model_run_message

Print the message to the console about which combination of the Time Series and LDA models is being run

Description

If desired, print a message at the beginning of every model combination stating the TS model and the LDA model being evaluated.
Usage

\[
\text{print_model_run_message}(\text{models}, \text{i}, \text{LDA\_models}, \text{control})
\]

Arguments

- **models**: A `data.frame` object returned from `expand_TS` that contains the combinations of LDA models, and formulas and nchangepoints used in the TS models.
- **i**: An integer index of the row to use from `models`.
- **LDA\_models**: A list of LDA models (class `LDA_set`, produced by `LDA_set`) or a singular LDA model (class `LDA`, produced by `LDA`).
- **control**: A list of parameters to control the fitting of the Time Series model including the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values not input assume defaults set by `TS\_control`. Of particular importance here is the logical-class element named `quiet`.

Value

- **NULL**.

Examples

```r
data(rodents)
document_term_table <- rodents\$document\_term\_table
document\_covariate\_table <- rodents\$document\_covariate\_table
LDAs <- LDA_set(document_term_table, topics = 2:3, nseeds = 2)
LDA\_models <- select\_LDA(LDAs)
weights <- document\_weights(document_term_table)
formulas <- c(~ 1, ~ newmoon)
nchangepoints <- 0:1
mods <- expand\_TS(LDA\_models, formulas, nchangepoints)
print\_model\_run\_message(mods, 1, LDA\_models, TS\_control())
```

---

**proposed\_step\_mods**

---

*Fit the chunk-level models to a time series, given a set of proposed change points within the ptMCMC algorithm*

Description

This function wraps around `TS\_memo` (optionally memoised `multinom\_TS`) to provide a simpler interface within the ptMCMC algorithm and is implemented within `propose\_step`.

Usage

```r
proposed\_step\_mods(prop\_changepts, inputs)
```
rho_lines

Arguments

prop_changepts  matrix of proposed change points across chains.
inputs          Class ptMCMC_inputs list, containing the static inputs for use within the ptMCMC algorithm.

Value

List of models associated with the proposed step, with an element for each chain.

Examples

data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)
data <- data[order(data[, "newmoon"]), ]
saves <- prep_saves(1, TS_control())
inputs <- prep_ptMCMC_inputs(data, gamma ~ 1, 1, "newmoon", weights, TS_control())
cpts <- prep_cpts(data, gamma ~ 1, 1, "newmoon", weights, TS_control())
i <- 1
pdist <- inputs$pdist
ntemps <- length(inputs$temps)
selection <- cbind(pdist$which_steps[i, ], 1:ntemps)
prop_changepts <- cpts$changepts
curr_changepts_s <- cpts$changepts[selection]
prop_changepts_s <- curr_changepts_s + pdist$steps[i, ]
if(all(is.na(prop_changepts_s))){
  prop_changepts_s <- NULL
}
prop_changepts[selection] <- prop_changepts_s
mods <- proposed_step_mods(prop_changepts, inputs)

rho_lines(mat)  Add change point location lines to the time series plot

Description

Adds vertical lines to the plot of the time series of fitted proportions associated with the change points of interest.

Usage

rho_lines(spec_rhos)
Arguments

spec_rhos numeric vector indicating the locations along the x axis where the specific change points being used are located.

Examples

```r
data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)
TSmod <- TS(data, gamma ~ 1, nchangepoints = 1, "newmoon", weights)
pred_gamma_TS_plot(TSmod)
rho_lines(200)
```

rodents  Portal rodent data

Description

An example LDATS dataset, functionally that used in Christensen et al. (2018). The data are counts of 21 rodent species across 436 sampling events, with the count being the total number observed across 8 50 m x 50 m plots, each sampled using 49 live traps (Brown 1998, Ernest et al. 2016).

Usage

rodents

Format

A list of two `data.frame`-class objects with rows corresponding to documents (sampling events). One element is the document term table (called `document_term_table`), which contains counts of the species (terms) in each sample (document), and the other is the document covariate table (called `document_covariate_table`) with columns of covariates (newmoon number, sin and cos of the fraction of the year).

Source

https://github.com/weecology/PortalData/tree/master/Rodents
References


select_LDA

Select the best LDA model(s) for use in time series

Description

Select the best model(s) of interest from an LDA_set object, based on a set of user-provided functions. The functions default to choosing the model with the lowest AIC value.

Usage

select_LDA(LDA_models = NULL, control = list())

Arguments

LDA_models An object of class LDA_set produced by LDA_set.
control A list of parameters to control the running and selecting of LDA models. Values not input assume default values set by LDA_set_control. Values for running the LDAs replace defaults in (LDAcontrol, see LDA (but if seed is given, it will be overwritten; use iseed instead).

Value

A reduced version of LDA_models that only includes the selected LDA model(s). The returned object is still an object of class LDA_set.

Examples

data(rodents)
da_data <- rodents$document_term_table
r_LDA <- LDA_set(lda_data, topics = 2, nseeds = 2)
select_LDA(r_LDA)
select_TS

Select the best Time Series model

Description

Select the best model of interest from an TS_on_LDA object generated by TS_on_LDA, based on a set of user-provided functions. The functions default to choosing the model with the lowest AIC value.

Presently, the set of functions should result in a singular selected model. If multiple models are chosen via the selection, only the first is returned.

Usage

select_TS(TS_models, control = list())

Arguments

TS_models An object of class TS_on_LDA produced by TS_on_LDA.
control A list of parameters to control the fitting of the Time Series model including the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values not input assume defaults set by TS_control.

Value

A reduced version of TS_models that only includes the selected TS model. The returned object is a single TS model object of class TS_fit.

Examples

data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDAs <- LDA_set(document_term_table, topics = 2:3, nseeds = 2)
LDA_models <- select_LDA(LDAs)
weights <- document_weights(document_term_table)
formulas <- c(~ 1, ~ newmoon)
mods <- TS_on_LDA(LDA_models, document_covariate_table, formulas,
    nchangepoints = 0:1, timename = "newmoon", weights)
select_TS(mods)
set_gamma_colors

Prepare the colors to be used in the gamma time series

Description

Based on the inputs, create the set of colors to be used in the time series of the fitted gamma (topic proportion) values.

Usage

set_gamma_colors(x, cols = NULL, option = "D", alpha = 1)

Arguments

x
Object of class TS_fit, fit by TS.

cols
Colors to be used to plot the time series of fitted topic proportions.

option
A character string indicating the color option from viridis to use if "cols == NULL". Four options are available: "magma" (or "A"), "inferno" (or "B"), "plasma" (or "C"), "viridis" (or "D", the default option) and "cividis" (or "E").

alpha
Numeric value [0,1] that indicates the transparency of the colors used. Supported only on some devices, see rgb.

Value

Vector of character hex codes indicating colors to use.

Examples

data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)
TSmod <- TS(data, gamma ~ 1, nchangepoints = 1, "newmoon", weights)
set_gamma_colors(TSmod)
**set_LDA_plot_colors**

Prepare the colors to be used in the LDA plots

**Description**

Based on the inputs, create the set of colors to be used in the LDA plots made by `plot.LDA_TS`.

**Usage**

```r
set_LDA_plot_colors(x, cols = NULL, option = "C", alpha = 0.8)
```

**Arguments**

- `x` Object of class LDA.
- `cols` Colors to be used to plot the topics. Any valid color values (e.g., see `colors`, `rgb`) can be input as with a standard plot. The default (cols = NULL) triggers use of `viridis` color options (see `option`).
- `option` A character string indicating the color option from `viridis` to use if `cols == NULL`. Four options are available: "magma" (or "A"), "inferno" (or "B"), "plasma" (or "C", the default option), "viridis" (or "D") and "cividis" (or "E").
- `alpha` Numeric value [0,1] that indicates the transparency of the colors used. Supported only on some devices, see `rgb`.

**Value**

vector of character hex codes indicating colors to use.

**Examples**

```r
data(rodents)
lda_data <- rodents$document_term_table
r_LDA <- LDA_set(lda_data, topics = 4, nseeds = 10)
set_LDA_plot_colors(r_LDA[[1]])
```
set_LDA_TS_plot_cols

Create the list of colors for the LDATS summary plot

Description
A default list generator function that produces the options for the colors controlling the panels of the LDATS summary plots, needed because the change point histogram panel should be in a different color scheme than the LDA and fitted time series model panels, which should be in a matching color scheme. See set_LDA_plot_colors, set_TS_summary_plot_cols, set_gamma_colors, and set_rho_hist_colors for specific details on usage.

Usage

set_LDA_TS_plot_cols(rho_cols = NULL, rho_option = "D", rho_alpha = 0.4, gamma_cols = NULL, gamma_option = "C", gamma_alpha = 0.8)

Arguments

rho_cols Colors to be used to plot the histograms of change points. Any valid color values (e.g., see colors, rgb) can be input as with a standard plot. The default (rho_cols = NULL) triggers use of viridis color options (see rho_option).
rho_option A character string indicating the color option from viridis to use if 'rho_cols == NULL'. Four options are available: "magma" (or "A"), "inferno" (or "B"), "plasma" (or "C"), "viridis" (or "D", the default option) and "cividis" (or "E").
rho_alpha Numeric value [0,1] that indicates the transparency of the colors used. Supported only on some devices, see rgb.
gamma_cols Colors to be used to plot the LDA topic proportions, time series of observed topic proportions, and time series of fitted topic proportions. Any valid color values (e.g., see colors, rgb) can be input as with a standard plot. The default (gamma_cols = NULL) triggers use of viridis color options (see gamma_option).
gamma_option A character string indicating the color option from viridis to use if gamma_cols == NULL'. Four options are available: "magma" (or "A"), "inferno" (or "B"), "plasma" (or "C", the default option), "viridis" (or "D") and "cividis" (or "E").
gamma_alpha Numeric value [0,1] that indicates the transparency of the colors used. Supported only on some devices, see rgb.

Value

list of elements used to define the colors for the two panels of the summary plot, as generated simply using set_LDA_TS_plot_cols. cols has two elements: LDA and TS, each corresponding the set of plots for its stage in the full model. LDA contains entries cols and options (see set_LDA_plot_colors). TS contains two entries, rho and gamma, each corresponding to the related panel, and each containing default values for entries named cols, option, and alpha (see set_TS_summary_plot_cols, set_gamma_colors, and set_rho_hist_colors).
```r
set_rho_hist_colors

Prepare the colors to be used in the change point histogram

Description
Based on the inputs, create the set of colors to be used in the change point histogram.

Usage

```r
defualt

```r
set_rho_hist_colors(x = NULL, cols = NULL, option = "D", alpha = 1)
```

Arguments

- `x` matrix of change point locations (element rhos) from an object of class TS_fit, fit by TS.
- `cols` Colors to be used to plot the histograms of change points. Any valid color values (e.g., see colors, rgb) can be input as with a standard plot. The default (rho_cols = NULL) triggers use of viridis color options (see rho_option).
- `option` A character string indicating the color option from viridis to use if "cols == NULL". Four options are available: "magma" (or "A"), "inferno" (or "B"), "plasma" (or "C"), "viridis" (or "D", the default option) and "cividis" (or "E").
- `alpha` Numeric value [0,1] that indicates the transparency of the colors used. Supported only on some devices, see rgb.

Value

Vector of character hex codes indicating colors to use.

Examples

```r
data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)
TSmod <- TS(data, gamma ~ 1, nchangepoints = 1, "newmoon", weights)
set_rho_hist_colors(TSmod$rhos)
```
set_TS_summary_plot_cols

Create the list of colors for the TS summary plot

Description
A default list generator function that produces the options for the colors controlling the panels of the TS summary plots, so needed because the panels should be in different color schemes. See set_gamma_colors and set_rhoh_hist_colors for specific details on usage.

Usage
set_TS_summary_plot_cols(rho_cols = NULL, rho_option = "D", rho_alpha = 0.4, gamma_cols = NULL, gamma_option = "C", gamma_alpha = 0.8)

Arguments
- rho_cols: Colors to be used to plot the histograms of change points. Any valid color values (e.g., see colors, rgb) can be input as with a standard plot. The default (rho_cols = NULL) triggers use of viridis color options (see rho_option).
- rho_option: A character string indicating the color option from viridis to use if rho_cols == NULL. Four options are available: "magma" (or "A"), "inferno" (or "B"), "plasma" (or "C"), "viridis" (or "D", the default option) and "cividis" (or "E").
- rho_alpha: Numeric value [0,1] that indicates the transparency of the colors used. Supported only on some devices, see rgb.
- gamma_cols: Colors to be used to plot the LDA topic proportions, time series of observed topic proportions, and time series of fitted topic proportions. Any valid color values (e.g., see colors, rgb) can be input as with a standard plot. The default (gamma_cols = NULL) triggers use of viridis color options (see gamma_option).
- gamma_option: A character string indicating the color option from viridis to use if gamma_cols == NULL. Four options are available: "magma" (or "A"), "inferno" (or "B"), "plasma" (or "C"), "viridis" (or "D", the default option) and "cividis" (or "E").
- gamma_alpha: Numeric value [0,1] that indicates the transparency of the colors used. Supported only on some devices, see rgb.

Value
list of elements used to define the colors for the two panels. Contains two elements rho and gamma, each corresponding to the related panel, and each containing default values for entries named cols, option, and alpha.

Examples
set_TS_summary_plot_cols()
Simulate LDA data from an LDA structure given parameters

Description

For a given set of parameters alpha and Beta and document-specific total word counts, simulate a document-by-term matrix. Additional structuring variables (the numbers of topics (k), documents (M), terms (V)) are inferred from input objects.

Usage

`sim_LDA_data(N, Beta, alpha = NULL, Theta = NULL, seed = NULL)`

Arguments

- **N**: A vector of document sizes (total word counts). Must be integer conformable. Is used to infer the total number of documents.
- **Beta**: Matrix of categorical distribution parameters defining terms within topics. Dimension: k x V (number of topics x number of terms). Used to infer both (k) and (V). Must be non-negative and sum to 1 within topics.
- **alpha**: Single positive numeric value for the Dirichlet distribution parameter defining topics within documents. To specifically define document topic probabilities, use Theta.
- **Theta**: Matrix of probabilities defining topics within documents. Dimension: M x k (documents x topics). Must be non-negative and sum to 1 within documents. To generally define document topic probabilities, use alpha.
- **seed**: Input to `set.seed`.

Value

A document-by-term matrix of counts (dim: M x V).

Examples

```r
N <- c(10, 22, 15, 31)
alpha <- 1.2
Beta <- matrix(c(0.1, 0.1, 0.8, 0.2, 0.6, 0.2), 2, 3, byrow = TRUE)
sim_LDA_data(N, Beta, alpha = alpha)
Theta <- matrix(c(0.2, 0.8, 0.2, 0.5, 0.9, 0.1), 4, 2, byrow = TRUE)
sim_LDA_data(N, Beta, Theta = Theta)
```
Simulate LDA_TS data from LDA and TS model structures and parameters

Description

For a given set of covariates $X$; parameters $\beta$, $\eta$, $\rho$, and $err$; and document-specific time stamps $tD$ and lengths $N$, simulate a document-by-topic matrix. Additional structuring variables (the numbers of topics ($k$), terms ($V$), documents ($M$), segments ($S$), and covariates per segment ($C$)) are inferred from input objects.

Usage

```
sim_LDA_TS_data(N, Beta, X, Eta, rho, tD, err = 0, seed = NULL)
```

Arguments

- **N**: A vector of document sizes (total word counts). Must be integer conformable. Is used to infer the total number of documents.
- **Beta**: matrix of categorical distribution parameters defining terms within topics. Dimension: $k \times V$ (number of topics x number of terms). Used to infer both ($k$) and ($V$). Must be non-negative and sum to 1 within topics.
- **X**: matrix of covariates, dimension $M$ (number of documents) x $C$ (number of covariates, including the intercept) (a.k.a the design matrix).
- **Eta**: matrix of regression parameters across the segments, dimension: $SC$ (number of segments x number of covariates, including the intercept) x $k$ (number of topics).
- **rho**: Vector of integer-conformable time locations of changepoints or NULL if no changepoints. Used to determine the number of segments. Must exist within the bounds of the times of the documents, $tD$.
- **tD**: Vector of integer-conformable times of the documents. Must be of length $M$ (as determined by $X$).
- **err**: Additive error on the link-scale. Must be a non-negative numeric value. Default value of 0 indicates no error.
- **seed**: Input to `set.seed`.

Value

A document-by-term matrix of counts (dim: $M \times V$).

Examples

```
N <- c(10, 22, 15, 31)
tD <- c(1, 3, 4, 6)
rho <- 3
X <- cbind(rep(1, 4), 1:4)
```
sim_TS_data

Simulate TS data from a TS model structure given parameters

description

For a given set of covariates X; parameters Eta, rho, and err; and document-specific time stamps tD, simulate a document-by-topic matrix. Additional structuring variables (numbers of topics (k), documents (M), segments (S), and covariates per segment (C)) are inferred from input objects.

usage

sim_TS_data(X, Eta, rho, tD, err = 0, seed = NULL)

arguments

X matrix of covariates, dimension M (number of documents) x C (number of covariates, including the intercept) (a.k.a. the design matrix).
Eta matrix of regression parameters across the segments, dimension: SC (number of segments x number of covariates, including the intercept) x k (number of topics).
rho Vector of integer-conformable time locations of changepoints or NULL if no changepoints. Used to determine the number of segments. Must exist within the bounds of the times of the documents, tD.
tD Vector of integer-conformable times of the documents. Must be of length M (as determined by X).
err Additive error on the link-scale. Must be a non-negative numeric value. Default value of 0 indicates no error.
seed Input to set.seed.

value

A document-by-topic matrix of probabilities (dim: M x k).

examples

tD <- c(1, 3, 4, 6)
rho <- 3
X <- cbind(rep(1, 4), 1:4)
Eta <- cbind(c(0.5, 0.3, 0.9, 0.5), c(1.2, 1.1, 0.1, 0.5))
sim_TS_data(X, Eta, rho, tD, err = 1)
softmax

Calculate the softmax of a vector or matrix of values

Description

Calculate the softmax (normalized exponential) of a vector of values or a set of vectors stacked rowwise.

Usage

softmax(x)

Arguments

x numeric vector or matrix

Value

The softmax of x.

Examples

dat <- matrix(runif(100, -1, 1), 25, 4)
softmax(dat)
softmax(dat[,1])

step_chains

Conduct a within-chain step of the ptMCMC algorithm

Description

This set of functions steps the chains forward one iteration of the within-chain component of the ptMCMC algorithm. `step_chains` is the main function, comprised of a proposal (made by `prop_step`), an evaluation of that proposal (made by `eval_step`), and then an update of the configuration (made by `take_step`).

This set of functions was designed to work within TS and specifically `est_changepoints`. They are still hardcoded to do so, but have the capacity to be generalized to work with any estimation via ptMCMC with additional coding work.
Usage

step_chains(i, cpts, inputs)

propose_step(i, cpts, inputs)

eval_step(i, cpts, prop_step, inputs)

take_step(cpts, prop_step, accept_step)

Arguments

i integer iteration index.

cpts matrix of change point locations across chains.

inputs Class ptMCMC_inputs list, containing the static inputs for use within the ptMCMC algorithm.

prop_step Proposed step output from propose_step.

accept_step logical indicator of acceptance of each chain’s proposed step.

Details

For each iteration of the ptMCMC algorithm, all of the chains have the potential to take a step. The possible step is proposed under a proposal distribution (here for change points we use a symmetric geometric distribution), the proposed step is then evaluated and either accepted or not (following the Metropolis-Hastings rule; Metropolis, et al. 1953, Hasting 1960, Gupta et al. 2018), and then accordingly taken or not (the configurations are updated).

Value

step_chains: list of change points, log-likelihoods, and logical indicators of acceptance for each chain.

propose_step: list of change points and log-likelihood values for the proposal.

eval_step: logical vector indicating if each chain’s proposal was accepted.

take_step: list of change points, log-likelihoods, and logical indicators of acceptance for each chain.

References


Examples

data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)
data <- data[nrow(data,"newmoon"),]
saves <- prep_saves(1, TS_control())
inputs <- prep_ptMCMC_inputs(data, gamma ~ 1, 1, "newmoon", weights, TS_control())
cpts <- prep_cpts(data, gamma ~ 1, 1, "newmoon", weights, TS_control())
ids <- prep_ids(TS_control())
for(i in 1:TS_control()$nit){
    steps <- step_chains(i, cpts, inputs)
    swaps <- swap_chains(steps, inputs, ids)
    saves <- update_saves(i, saves, steps, swaps)
    cpts <- update_cpts(cpts, swaps)
    ids <- update_ids(ids, swaps)
}

# within step_chains()
cpts <- prep_cpts(data, gamma ~ 1, 1, "newmoon", weights, TS_control())
i <- 1
prop_step <- propose_step(i, cpts, inputs)
accept_step <- eval_step(i, cpts, prop_step, inputs)
take_step(cpts, prop_step, accept_step)

summarize_etas

Summarize the regressor (eta) distributions

Description

summarize_etas calculates summary statistics for each of the chunk-level regressors.

measure_ets_vcov generates the variance-covariance matrix for the regressors.

Usage

summarize_etas(etas, control = list())

measure_ets_vcov(etas)
Argument

**etas**
Matrix of regressors (columns) across iterations of the ptMCMC (rows), as returned from `est_regressors`.

**control**
A list of parameters to control the fitting of the Time Series model including the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values not input assume defaults set by `TS_control`.

Value

`summarize_etas`: table of summary statistics for chunk-level regressors including mean, median, mode, posterior interval, standard deviation, MCMC error, autocorrelation, and effective sample size for each regressor.

`measure_eta_vcov`: variance-covariance matrix for chunk-level regressors.

Examples

```r
etas <- matrix(rnorm(100), 50, 2)
summarize_etas(etas)
measure_eta_vcov(etas)
```

summarize_rhos **Summarize the rho distributions**

Description

`summarize_rhos` calculates summary statistics for each of the change point locations.

`measure_rho_vcov` generates the variance-covariance matrix for the change point locations.

Usage

```r
summarize_rhos(rhos, control = list())
measure_rho_vcov(rhos)
```

Arguments

**rhos**
Matrix of change point locations (columns) across iterations of the ptMCMC (rows) or NULL if no change points are in the model, as returned from `est_changepoints`.

**control**
A list of parameters to control the fitting of the Time Series model including the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values not input assume defaults set by `TS_control`.
swap_chains

Value

summarize_rhos: table of summary statistics for change point locations including mean, median, mode, posterior interval, standard deviation, MCMC error, autocorrelation, and effective sample size for each change point location.

measure_rho_vcov: variance-covariance matrix for change point locations.

Examples

rhos <- matrix(sample(80:100, 100, TRUE), 50, 2)
summarize_rhos(rhos)
measure_rho_vcov(rhos)

swap_chains

Conduct a set of among-chain swaps for the ptMCMC algorithm

Description

This function handles the among-chain swapping based on temperatures and likelihood differentials.

This function was designed to work within TS and specifically est_changepoints. It is still hard-coded to do so, but has the capacity to be generalized to work with any estimation via ptMCMC with additional coding work.

Usage

swap_chains(chainsin, inputs, ids)

Arguments

chainsin  Chain configuration to be evaluated for swapping.
inputs   Class ptMCMC_inputs list, containing the static inputs for use within the ptMCMC algorithm.
ids  The vector of integer chain ids.

Details

The ptMCMC algorithm couples the chains (which are taking their own walks on the distribution surface) through "swaps", where neighboring chains exchange configurations (Geyer 1991, Falcioni and Deem 1999) following the Metropolis criterion (Metropolis et al. 1953). This allows them to share information and search the surface in combination (Earl and Deem 2005).

Value

list of updated change points, log-likelihoods, and chain ids, as well as a vector of acceptance indicators for each swap.
**References**


**Examples**

```r
data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)
data <- data[order(data[, "newmoon"]), ]
saves <- prep_saves(1, TS_control())
inputs <- prep_ptMCMC_inputs(data, gamma ~ 1, 1, "newmoon", weights, TS_control())
cpts <- prep_cpts(data, gamma ~ 1, 1, "newmoon", weights, TS_control())
ids <- prep_ids(TS_control())
for(i in 1:TS_control()$nit){
  steps <- step_chains(i, cpts, inputs)
  swaps <- swap_chains(steps, inputs, ids)
  saves <- update_saves(i, saves, steps, swaps)
  cpts <- update_cpts(cpts, swaps)
  ids <- update_ids(ids, swaps)
}
```

---

**trace_plot**

Produce the trace plot panel for the TS diagnostic plot of a parameter

**Description**

Produce a trace plot for the parameter of interest (rho or eta) as part of TS_diagnostics_plot. A horizontal line is added to show the median of the posterior.
Usage

trace_plot(x, ylab = "parameter value")

Arguments

x Vector of parameter values drawn from the posterior distribution, indexed to the iteration by the order of the vector.

ylab character value used to label the y axis.

Value

NULL.

Examples

trace_plot(rnorm(100, 0, 1))

---

TS Conduct a single multinomial Bayesian Time Series analysis

Description

This is the main interface function for the LDATS application of Bayesian change point Time Series analyses (Christensen et al. 2018), which extends the model of Western and Kleykamp (2004; see also Ruggieri 2013) to multinomial (proportional) response data using softmax regression (Ripley 1996, Venables and Ripley 2002, Bishop 2006) using a generalized linear modeling approach (McCullagh and Nelder 1989). The models are fit using parallel tempering Markov Chain Monte Carlo (ptMCMC) methods (Earl and Deem 2005) to locate change points and neural networks (Ripley 1996, Venables and Ripley 2002, Bishop 2006) to estimate regressors.

check_TS_inputs checks that the inputs to TS are of proper classes for a full analysis.

Usage

TS(data, formula = gamma ~ 1, nchangepoints = 0, timename = "time", weights = NULL, control = list())

check_TS_inputs(data, formula = gamma ~ 1, nchangepoints = 0, timename = "time", weights = NULL, control = list())
Arguments

- **data**: `data.frame` including [1] the time variable (indicated in `timename`), [2] the predictor variables (required by `formula`) and [3], the multinomial response variable (indicated in `formula`) as verified by `check_timename` and `check_formula`. Note that the response variables should be formatted as a `data.frame` object named as indicated by the response entry in the `control` list, such as gamma for a standard TS analysis on LDA output. See Examples.

- **formula**: `formula` defining the regression between relationship the change points. Any predictor variable included must also be a column in `data` and any (multinomial) response variable must be a set of columns in `data`, as verified by `check_formula`.

- **nchangepoints**: integer corresponding to the number of change points to include in the model. 0 is a valid input (corresponding to no change points, so a singular time series model), and the current implementation can reasonably include up to 6 change points. The number of change points is used to dictate the segmentation of the time series into chunks fit with separate models dictated by `formula`.

- **timename**: character element indicating the time variable used in the time series. Defaults to "time". The variable must be integer-conformable or a `Date`. If the variable named is a `Date`, the input is converted to an integer, resulting in the timestep being 1 day, which is often not desired behavior.

- **weights**: Optional class numeric vector of weights for each document. Defaults to `NULL`, translating to an equal weight for each document. When using `multinom_TS` in a standard LDATS analysis, it is advisable to weight the documents by their total size, as the result of `LDA` is a matrix of proportions, which does not account for size differences among documents. For most models, a scaling of the weights (so that the average is 1) is most appropriate, and this is accomplished using `document_weights`.

- **control**: A list of parameters to control the fitting of the Time Series model including the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values not input assume defaults set by `TS_control`.

Value

- **TS**: `TS_fit`-class list containing the following elements, many of which are hidden for printing, but are accessible:

  - **data**: data input to the function.
  - **formula**: formula input to the function.
  - **nchangepoints**: nchangepoints input to the function.
  - **weights**: weights input to the function.
  - **control**: control input to the function.
  - **lls**: Iteration-by-iteration logLik values for the full time series fit by `multinom_TS`.
  - **rhos**: Iteration-by-iteration change point estimates from `est_changepoints`.
  - **etas**: Iteration-by-iteration marginal regressor estimates from `est_regressors`, which have been unconditioned with respect to the change point locations.
ptMCMC_diagnostics  ptMCMC diagnostics, see diagnose_ptMCMC
rho_summary  Summary table describing rhos (the change point locations), see summarize_rhos.
rho_vcov  Variance-covariance matrix for the estimates of rhos (the change point locations), see measure_rho_vcov.

eta_summary  Summary table describing etas (the regressors), see summarize_etas.
eta_vcov  Variance-covariance matrix for the estimates of etas (the regressors), see measure_eta_vcov.

logLik  Across-iteration average of log-likelihoods (lls).
nparams  Total number of parameters in the full model, including the change point locations and regressors.
deviance  Penalized negative log-likelihood, based on logLik and nparams.

check_TS_inputs: An error message is thrown if any input is not proper, else NULL.

References


Examples

```r
data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)

TSmod <- TS(data, gamma ~ 1, nchangepoints = 1, "newmoon", weights)

check_TS_inputs(data, timename = "newmoon")
```
**TS_control**  
*Create the controls list for the Time Series model*

**Description**

This function provides a simple creation and definition of a list used to control the time series model fit occurring within `TS`.

**Usage**

```r
TS_control(memoise = TRUE, response = "gamma", lambda = 0,
           measurer = AIC, selector = min, ntemps = 6,
           penultimate_temp = 2^6, ultimate_temp = 1e+10, q = 0,
           nit = 10000, magnitude = 12, quiet = FALSE, burnin = 0,
           thin_frac = 1, summary_prob = 0.95, seed = NULL)
```

**Arguments**

- `memoise` logical indicator of whether the multinomial functions should be memoised (via `memoise`). Memoisation happens to both `multinom_TS` and `multinom_TS_chunk`.
- `response` character element indicating the response variable used in the time series.
- `lambda` numeric "weight" decay term used to set the prior on the regressors within each chunk-level model. Defaults to 0, corresponding to a fully vague prior.
- `measurer`, `selector` Function names for use in evaluation of the TS models. `measurer` is used to create a value for each model and `selector` operates on the values to choose the model.
- `ntemps` integer number of temperatures (chains) to use in the ptMCMC algorithm.
- `penultimate_temp` Penultimate temperature in the ptMCMC sequence.
- `ultimate_temp` Ultimate temperature in the ptMCMC sequence.
- `q` Exponent controlling the ptMCMC temperature sequence from the focal chain (reference with temperature = 1) to the penultimate chain. 0 (default) implies a geometric sequence. 1 implies squaring before exponentiating.
- `nit` integer number of iterations (steps) used in the ptMCMC algorithm.
- `magnitude` Average magnitude (defining a geometric distribution) for the proposed step size in the ptMCMC algorithm.
- `quiet` logical indicator of whether the model should run quietly (if `FALSE`, a progress bar and notifications are printed).
- `burnin` integer number of iterations to remove from the beginning of the ptMCMC algorithm.
- `thin_frac` Fraction of iterations to retain, must be \((0, 1]\), and the default value of 1 represents no thinning.
**summary_prob**  Probability used for summarizing the posterior distributions (via the highest posterior density interval, see **HPDinterval**).

**seed**  Input to `set.seed` for replication purposes.

**Value**

`list`, with named elements corresponding to the arguments.

**Examples**

```r
TS_control()
```

---

**TS_diagnostics_plot**  *Plot the diagnostics of the parameters fit in a TS model*

**Description**

Plot 4-panel figures (showing trace plots, posterior ECDF, posterior density, and iteration autocorrelation) for each of the parameters (change point locations and regressors) fitted within a multinomial time series model (fit by **TS**).

- `eta_diagnostics_plots` creates the diagnostic plots for the regressors (etas) of a time series model.

- `rho_diagnostics_plots` creates the diagnostic plots for the change point locations (rho) of a time series model.

**Usage**

```r
TS_diagnostics_plot(x, interactive = TRUE)
eta_diagnostics_plots(x, interactive)
rho_diagnostics_plots(x, interactive)
```

**Arguments**

- **x**  Object of class **TS_fit**, generated by **TS** to have its diagnostics plotted.

- **interactive**  logical input, should be code `TRUE` unless testing.

**Value**

`NULL`.
**Examples**

```r
data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)
TSmod <- TS(data, gamma ~ 1, nchangepoints = 1, "newmoon", weights)
TS_diagnostics_plot(TSmod)
```

---

**TS_on_LDA**

*Conduct a set of Time Series analyses on a set of LDA models*

**Description**

This is a wrapper function that expands the main Time Series analyses function (TS) across the LDA models (estimated using LDA or LDA_set and the Time Series models, with respect to both continuous time formulas and the number of discrete changepoints. This function allows direct passage of the control parameters for the parallel tempering MCMC through to the main Time Series function, TS, via the ptMCMC_controls argument.

**Usage**

```r
TS_on_LDA(LDA_models, document_covariate_table, formulas = ~1,
nchangepoints = 0, timename = "time", weights = NULL,
control = list())
```

**Arguments**

- `LDA_models` List of LDA models (class LDA_set, produced by LDA_set) or a singular LDA model (class LDA, produced by LDA).
- `document_covariate_table` Document covariate table (rows: documents, columns: time index and covariate options). Every model needs a covariate to describe the time value for each document (in whatever units and whose name in the table is input in timename) that dictates the application of the change points. In addition, all covariates named...
within specific models in formula must be included. Must be a conformable to
a data table, as verified by check_document_covariate_table.

formulas Vector of formula(s) for the continuous (non-change point) component of the
time series models. Any predictor variable included in a formula must also be
a column in the document_covariate_table. Each element (formula) in the
vector is evaluated for each number of change points and each LDA model.

nchangepoints Vector of integers corresponding to the number of change points to include in
the time series models. 0 is a valid input corresponding to no change points (i.e.,
a singular time series model), and the current implementation can reasonably
include up to 6 change points. Each element in the vector is the number of
change points used to segment the data for each formula (entry in formulas)
component of the TS model, for each selected LDA model.

timename character element indicating the time variable used in the time series. Defaults
to "time". The variable must be integer-conformable or a Date. If the variable
named is a Date, the input is converted to an integer, resulting in the timestep
being 1 day, which is often not desired behavior.

weights Optional class numeric vector of weights for each document. Defaults to NULL,
translating to an equal weight for each document. When using multinom_TS in
a standard LDATS analysis, it is advisable to weight the documents by their total
size, as the result of LDA is a matrix of proportions, which does not account for
size differences among documents. For most models, a scaling of the weights
(so that the average is 1) is most appropriate, and this is accomplished using
document_weights.

control A list of parameters to control the fitting of the Time Series model including
the parallel tempering Markov Chain Monte Carlo (ptMCMC) controls. Values
not input assume defaults set by TS_control.

Value

TS_on_LDA: TS_on_LDA-class list of results from TS applied for each model on each LDA model
input.

check_TS_inputs: An error message is thrown if any input is not proper, else NULL.

Examples

data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDAs <- LDA_set(document_term_table, topics = 2:3, nseeds = 2)
LDA_models <- select_LDA(LDAs)
weights <- document_weights(document_term_table)
formulas <- c(~ 1, ~ newmoon)
mods <- TS_on_LDA(LDA_models, document_covariate_table, formulas,
nchangepoints = 0:1, timename = "newmoon", weights)
Create the summary plot for a TS fit to an LDA model

Description

Produces a two-panel figure of [1] the change point distributions as histograms over time and [2] the time series of the fitted topic proportions over time, based on a selected set of change point locations.

pred_gamma_TS_plot produces a time series of the fitted topic proportions over time, based on a selected set of change point locations.

rho_hist: make a plot of the change point distributions as histograms over time.

Usage

TS_summary_plot(x, cols = set_TS_summary_plot_cols(), bin_width = 1,
  xname = NULL, border = NA, selection = "median", LDATS = FALSE)

pred_gamma_TS_plot(x, selection = "median", cols = set_gamma_colors(x),
  xname = NULL, together = FALSE, LDATS = FALSE)

rho_hist(x, cols = set_rhos_hist_colors(x$rhos), bin_width = 1,
  xname = NULL, border = NA, together = FALSE, LDATS = FALSE)

Arguments

x Object of class TS_fit produced by TS.

cols list of elements used to define the colors for the two panels, as generated simply using set_TS_summary_plot_cols. Has two elements rho and gamma, each corresponding to the related panel, and each containing default values for entries named cols, option, and alpha. See set_gamma_colors and set_rhos_hist_colors for details on usage.

bin_width Width of the bins used in the histograms, in units of the x-axis (the time variable used to fit the model).

xname Label for the x-axis in the summary time series plot. Defaults to NULL, which results in usage of the timename element of the control list (held in control$TS_control$timename). To have no label printed, set xname = "".

border Border for the histogram, default is NA.

selection Indicator of the change points to use. Currently only defined for "median" and "mode".

LDATS logical indicating if the plot is part of a larger LDATS plot output.

together logical indicating if the subplots are part of a larger LDA plot output.

Value

NULL.
Examples

data(rodents)
document_term_table <- rodents$document_term_table
document_covariate_table <- rodents$document_covariate_table
LDA_models <- LDA_set(document_term_table, topics = 2)[[1]]
data <- document_covariate_table
data$gamma <- LDA_models@gamma
weights <- document_weights(document_term_table)
TSmod <- TS(data, gamma ~ 1, nchangepoints = 1, "newmoon", weights)
TS_summary_plot(TSmod)
pred_gamma_TS_plot(TSmod)
rho_hist(TSmod)

verify_changepoint_locations

Verify the change points of a multinomial time series model

Description

Verify that a time series can be broken into a set of chunks based on input change points.

Usage

verify_changepoint_locations(data, changepoints = NULL, timename = "time")

Arguments

data Class data.frame object including the predictor and response variables.

cangepoints Numeric vector indicating locations of the change points. Must be conformable to integer values.

timename character element indicating the time variable used in the time series. Defaults to "time". The variable must be integer-conformable or a Date. If the variable named is a Date, the input is converted to an integer, resulting in the timestep being 1 day, which is often not desired behavior.

Value

Logical indicator of the check passing TRUE or failing FALSE.
Examples

data(rodents)
dtt <- rodents$document_term_table
lda <- LDA_set(dtt, 2, 1, list(quiet = TRUE))
dct <- rodents$document_covariate_table
dct$gamma <- lda[[1]]@gamma
verify_changepoint_locations(dct, changepoints = 100,
                             timename = "newmoon")
Index

*Topic datasets
  jornada, 21
  rodents, 64

*Topic package
  LDATS, 22

acf, 4
AIC, 4
AICc, 4
autocorr_plot, 4

character, 36
check_changepoints, 5, 35
check_control, 5, 58
check_document_covariate_table, 6, 26, 58, 87
check_document_term_table, 7, 7, 15, 24, 26
check_formula, 7, 8, 17, 18, 34, 41, 50, 82
check_formulas, 8
check_LDA_models, 9
check_LDA_set_inputs (LDA_set), 23
check_LDA_TS_inputs (LDA_TS), 25
check_multinom_TS_inputs (multinom_TS), 34
check_nchangepoints, 10
check_seeds, 10
check_timename, 8, 11, 17, 18, 34, 41, 50, 82
check_topics, 11
check_TS_inputs (TS), 81
check_TS_on_LDA_inputs (TS_on_LDA), 86
check_weights, 12
colors, 46, 68–71
conform_LDA_TS_data (LDA_TS), 25
count_trips, 13, 14
diagnose_ptMCMC, 13, 14, 41, 83
document_weights, 15, 35, 50, 55
ecdf_plot, 16

est_changepoints, 13, 14, 16, 18, 41, 51, 55–57, 75, 78, 79, 82
est_regressors, 18, 41, 78, 82
eta_diagnostics_plots (TS_diagnostics_plot), 85
eval_step (step_chains), 75
expand_TS, 20, 43, 59, 62

formula, 7, 8, 17, 18, 20, 26, 34, 36, 41, 50, 82, 87

HPDinterval, 28, 85

iftrue, 21

jornada, 21

LDA, 9, 17, 18, 20, 23–26, 28, 29, 35, 36, 39, 41, 43, 45, 50, 52, 55, 58, 62, 65, 82, 86, 87
LDA_msg, 23
LDA_plot_bottom_panel (plot.LDA_VEM), 45
LDA_plot_top_panel (plot.LDA_VEM), 45
LDA_set, 9, 20, 23, 25, 27, 38–40, 43, 58, 62, 65, 86
LDA_set_control, 24, 24, 65
LDA_TS, 25, 27, 39, 40, 44, 59
LDA_TS_control, 26, 27
LDATS, 22
LDATS-package (LDATS), 22
logLik, 4, 30, 41, 82
logLik.LDA_VEM, 29
logLik.multinom_TS_fit, 30
logLik.TS_fit, 30
logsumexp, 31

measure_eta_vcov, 42, 83
measure_eta_vcov (summarize_etas), 77
measure_rho_vcov, 42, 83
measure_rho_vcov (summarize_rhos), 78
memoise, 28, 32, 84
memoise_fun, 32
messageq, 32
mirror_vcov, 33
modalvalue, 33
multinom, 30, 33, 34, 36
multinom_TS, 18, 28, 30, 34, 41, 62, 82, 84
multinom_TS_chunk, 28, 36, 37, 38, 84
normalize, 37
package_chunk_fits, 37
package_LDA_set, 38
package_LDA_TS, 39
package_TS, 40
package_TS_on_LDA, 42
plot, 43, 44, 46, 47
plot.LDA_set, 43
plot.LDA_TS, 44, 68
plot.LDA_VEM, 45
plot.TS_fit, 46
posterior_plot, 48
pred.gamma_TS_plot (TS_summary_plot), 88
prep_chunks, 48
prep_cpts, 49
prep_ids, 51
prep_LDA_control, 52
prep_pbar, 53
prep_proposal_dist, 54
prep_ptMCMC_inputs, 55
prep_saves, 56
prep_temp_sequence, 57
prep_TS_data, 58
print.LDA_TS, 59
print.TS_fit, 60
print.TS_on_LDA, 61
print_model_run_message, 61
process_saves (prep_saves), 56
propose_step, 62
propose_step (step_chains), 75
proposed_step_mods, 62
rgb, 46, 67–71
rho_diagnostics_plots (TS_diagnostics_plot), 85
rho_hist (TS_summary_plot), 88
rho_lines, 63
rodents, 64
select_LDA, 25, 27, 39, 40, 65
select_TS, 25, 27, 39, 40, 66
set.seed, 72–74
set.gamma_colors, 44, 47, 67, 69, 71, 88
set_LDA_plot_colors, 44, 68, 69
set_LDA_TS_plot_cols, 44, 69, 69
set_rho_hist_colors, 44, 47, 69, 70, 71, 88
set_TS_summary_plot_cols, 44, 47, 69, 71, 88
sim_LDA_data, 72
sim_LDA_TS_data, 73
sim_TS_data, 74
softmax, 75
step_chains, 75
summarize_etas, 42, 77, 83
summarize_rhos, 41, 78, 83
swap_chains, 79
take_step (step_chains), 75
trace_plot, 80
TS, 13, 14, 40, 42, 43, 46, 47, 51, 53, 55–58, 60, 67, 70, 75, 79, 81, 84–88
TS_control, 8, 17, 18, 35, 37, 36, 41, 50, 51, 53–55, 57, 58, 62, 66, 78, 82, 84, 87
TS_diagnostics_plot, 4, 16, 48, 80, 85
TS_on_LDA, 25, 27, 39, 40, 61, 66, 86
TS_summary_plot, 88
update_cpts (prep_cpts), 49
update_ids (prep_ids), 51
update_pbar (prep_pbar), 53
update_saves (prep_saves), 56
vcov, 33
verify_changepoint_locations, 35, 89
viridis, 46, 67–71