Package 'STL2018'

May 22, 2020

Type Package

Version 0.1.0

Date 2020-05-08

Title What the Package Does (Title Case)

Author Davis Berlind [aut, cre]

Maintainer Davis Berlind <davis.berlind@gmail.com>

Description This package is an implementation of the methods described in Steorts, R.C., Tancredi, A., Liseo, B. Generalized Bayesian Record Linkage and Regression with Exact Error Propagation, PSD (2018). The methods implemented in this package enable the user to recreate the experiments described in this paper, and allow for further experimentation with joint record linkage and linear regression tasks.

License What license is it under?

Encoding UTF-8

LazyData true

Import BB,
doParallel,
foreach,
magrittr,
Matrix,
mvnfast,
parallel,
pracma,
tcltk

RoxygenNote 7.1.0

R topics documented:

`alpha_metropolis` *Metropolis-Hastings update of distortion probability for ℓth field.*

Description

Implementation of the Metropolis-Hastings update step for the distortion probability α_ℓ. The default proposal distribution is a reflected random walk.

Usage

```
alpha_metropolis(current, linkage, field, thetas_l, a, b, proposal, drift)
```

Arguments

`current`	A float. The current value of α_ℓ in the Markov chain.
`linkage`	An integer vector. The cluster labels for each record.
`field`	A character vector. The values for the ℓ^{th} field of the records.
`thetas_l`	A matrix. The first column of `thetas_l` must contain all the possible values that the ℓ^{th} field can take (i.e. the unique values of `field`), and the second column must contain the corresponding frequencies of each value in the whole set of records.
`a`	A float. The α shape parameter of the beta prior for α_ℓ.
`b`	A float. The β shape parameter of the beta prior for α_ℓ.
`proposal`	A string. Specifies the proposal distribution to be used in generating a new value for α_ℓ. Currently only accepts "RRW" for reflected random walk.
`drift`	A float. A control parameter for tuning the variance of the proposal distribution. larger values will lead to less correlated proposals.

Value

The next value of α_ℓ in the Markov chain.

alpha_prior_tests *Tests for validity of* alpha_pior *argument.*

Description

Both recursiveRL and regressionRL require an alpha_pior argument that encodes the beta prior on the distortion probabilites. alpha_prior_tests runs tests to make sure alpha_pior is a valid data structure.

Usage

```
alpha_prior_tests(key_vars, alpha_prior)
```

Arguments

key_vars	A character vector. The names of the fields in the record data.
alpha_prior	A list of lists. A list of beta distribution shape parameters representing the prior for the distortion probabilities.

Value

Exception and error message if alpha_prior is invalid.

Details

alpha_pior must be a list of lists. Each list in alpha_pior must be named after an element of key_vars, and must contain two strictly positive numeric elements alpha and beta. Each element of alpha_pior can also include two additional parameters proposal and drift. proposal should be a string specifying the proposal distribution for each α_ℓ and drift should be a non-negative number specifying the variance of the proposal distribution. alpha_prior_tests runs tests to make sure alpha_prior is structured according to these standards.

See Also

alpha_metropolis for currently implemented values of proposal.

Other Prior Functions.: generate_alpha_prior()

Examples

```
# creating uniform prior for RL500

key_vars <- c("fname_c1", "lname_c1", "by", "bm", "bd")

alpha_prior <- list(fname_c1 = list(alpha = 1, beta = 1,
                                    proposal = "RRW", drift = 0.5),
                    lname_c1 = list(alpha = 1, beta = 1,
                                    proposal = "RRW", drift = 0.5),
                    by       = list(alpha = 1, beta = 1,
                                    proposal = "RRW", drift = 0.5),
                    bm       = list(alpha = 1, beta = 1,
                                    proposal = "RRW", drift = 0.5),
                    bd       = list(alpha = 1, beta = 1,
```

```
                                    proposal = "RRW", drift = 0.5))

alpha_prior_tests(key_vars, alpha_prior)
```

`beta_optim` *Optimization function for backsolving shape parameters for a beta prior.*

Description

Optimization function for backsolving shape parameters for a beta prior.

Usage

```
beta_optim(par, mn, var)
```

Arguments

`par`	Parameters to optimize over.
`mn`	A float. The mean condition of the beta prior. Must be strictly positive.
`var`	A float. The variance condition of the beta prior. Must be strictly positive.

Value

A vector of residuals.

`beta_solve` *Solve for the shape parameters of beta distribtution given mean and variance.*

Description

`beta_solve` backsolves for α and β of a beta distribution given a mean and variance.

Usage

```
beta_solve(mn, var)
```

Arguments

`mn`	A float. The mean condition of the beta prior. Must be strictly positive.
`var`	A float. The variance condition of the beta prior. Must be strictly positive.

Value

A list. (α, β)

Examples

```
beta_solve(mn = 2.6, var = 3.4)
```

`data_tests` *Title*

Description

Title

Usage

```
data_tests(data, key_vars)
```

Arguments

`key_vars`

`emp_dist` *Calculates empirical frequencies of each value in a collection of records.*

Description

`emp_dist` takes a data.frame of records `data` and `key_vars` a vector of colmun names, then returns a list of data.frames (one for each key variable), each containing a column of values and a column of corresponding empirical frequencies.

Usage

```
emp_dist(data, key_vars)
```

Arguments

`data`	A data.frame. The data containing the records.
`key_vars`	A character vector. The names of the columns from `data` to use as key variables.

Value

A list of data.frames. Each list contains an index of values each key variable can take on and the corresponding empirical frequency.

generate_alpha_prior *Generate prior for distortion probability α*

Description

Given a vector of key variable names, `generate_alpha_prior` prompts the user to enter the parameters of the prior on the distortion probability. The list that `generate_alpha_prior` can then directly be used as the `alpha_prior` argument in either `recursiveRL` or `regressionRL`.

Usage

```
generate_alpha_prior(key_vars)
```

Arguments

`key_vars`	A character vector. The names of the fields in the record data.

Value

A formatted list encoding the prior for α

See Also

Other Prior Functions.: `alpha_prior_tests()`

Examples

```
key_vars <- c("fname_c1", "lname_c1", "by", "bm", "bd")
alpha_prior <- generate_alpha_prior(key_vars)
```

get_alpha_proposals *List the implemented proposal distributions for the distortion probability α.*

Description

Returns a list of the proposal distributions that are currently implemented for generating new values of the distortion probability α in `alpha_metropolis`.

Usage

```
get_alpha_proposals(verbose = TRUE)
```

Arguments

`verbose`	A logical. If TRUE, the list of implemented proposal distributions will be printed

Value

A character vector of the implemented proposal distributions.

`hitmiss` *Likelihood for the ℓth field of the hit-and-miss model.*

Description

`hitmiss` gives the likelihood of the ℓ^{th} field of a record taking the value `v` given that the true value for the field is `v_true`, the empirical frequency of `v` in all of the records is $\theta_{v,\ell}$, and the distortion probability for the ℓ^{th} field is α_ℓ.

Usage

```
hitmiss(v, v_true, theta, alpha)
```

Arguments

`v`	A string. The potential value for the ℓ^{th} field of the record that `hitmiss` calculates the likelihood for.
`v_true`	A string. The true value of the ℓ^{th} field of the record.
`theta`	A float. The empirical frequency of `v` in the set of records.
`alpha`	A float. The distortion probability for the ℓ^{th} field.

Value

The likelihood of `v`.

Details

This function is used in the Metropolis-Hastings update step for the linkage structure Λ.

Given that the true value for the ℓ^{th} field is `v_true`, the empirical frequency of `v` in all of the records is $\theta_{v,\ell}$, and the distortion probability for the ℓ^{th} field is α_ℓ, then the likelihood of `v` is given by

$$(1-\alpha_\ell)\delta(v, v_{true}) + \alpha_\ell \theta_{v,\ell},$$

where $\delta(v, v_{true}) = 1$ if $v = v_{true}$, and $\delta(v, v_{true}) = 0$ otherwise.

See Also

Other Cluster Likelihood Functions: `log_distortion_cluster()`, `log_joint_lm_cluster()`, `log_linkage_cluster()`, `log_lm_cluster()`, `recursive_cluster()`

lambda_prior_tests *Tests for validity of* lambda_pior *argument.*

Description

Both recursiveRL and regressionRL require an lambda_pior argument that encodes the prior on the linkage structure Λ. lambda_prior_tests runs tests to make sure lambda_pior is a valid data structure.

Usage

```
lambda_prior_tests(lambda_prior)
```

Arguments

lambda_prior	A list. A list containing the type of prior to use for the linkage structure and the corresponding parameters for the prior.

Value

Exception and error message if lambda_prior is invalid.

Details

lambda_prior must include an element named "prior" that contains a string specifying the prior to be used for the linkage structure. Currently the values 'uniform' and 'PYP' are supported. If 'prior' = 'PYP', then lambda_prior must also contain two numeric arguments 'nu' and 'sigma'. See pyp_mean and pyp_var for more detail.

Examples

```
uni_prior <- list(prior = "uniform")
pyp_params <- pyp_solve(mn = 450, var = 500, N = 500)
pyp_prior <- list(prior = "PYP",
                  nu = pyp_params$nu,
                  sigma = pyp_params$sigma)

lapply(list(uni_prior, pyp_prior), lambda_prior_tests)
```

lgratio *Calculate* $\log \frac{\Gamma(a+b)}{\Gamma(a)}$.

Description

Calculate $\log \frac{\Gamma(a+b)}{\Gamma(a)}$.

Usage

```
lgratio(a, b)
```

Arguments

a, b	A float. $a > 0$ and $b > -a$.

Value

$\log \frac{\Gamma(a+b)}{\Gamma(a)}$

See Also

Other PYP Functions: pyp_mean(), pyp_optim(), pyp_solve(), pyp_var(), pyp_warnings()

---

log_distortion_cluster

Joint cluster likelihood for the ℓth field.

Description

Given all the values for the ℓ^{th} field of the observed records in field, as well as the cluster membership for each record in linkage, log_alpha_cluster returns the joint log-likelihood of all the clusters for the ℓ^{th} field.

Usage

```
log_distortion_cluster(linkage, field, alpha_l, thetas_l)
```

Arguments

linkage	An integer vector. The cluster labels for each record.
field	A character vector. The values for the ℓ^{th} field of the records.
alpha_l	A float. The distortion probability for the ℓ^{th} field.
thetas_l	A matrix. The first column of thetas_l must contain all the possible values that the ℓ^{th} field can take (i.e. the unique values of field), and the second column must contain the corresponding frequencies of each value in the whole set of records.

Value

The joint cluster log-likelihood for the ℓ^{th} field.

Details

This function is used in the Metropolis-Hastings update step for the distortion probabilities α.

See Also

recursive_cluster for the actual recursion formula implemented for this function.

Other Cluster Likelihood Functions: hitmiss(), log_joint_lm_cluster(), log_linkage_cluster(), log_lm_cluster(), recursive_cluster()

`log_joint_lm_cluster` *Joint conditional regression log-likelihood.*

Usage

```
log_joint_lm_cluster(linkage, model_data, beta, sigma_data, sigma_y, sigma_x)
```

Arguments

`linkage`	An integer vector. The cluster labels for each record.
`model_data`	A data.frame. The model.frame of the regression. The first column must contain the outcome variable y and the remaining columns must contain the model covariates X.
`beta`	A numeric vector. The regression coefficients. Can also be a single float in the case of univariate regression.
`sigma_data`	A matrix. The variance-covariance matrix of the covariates. Must be symetric positive-definite. Can also be a strictly positive float in the case of univariate regression.
`sigma_y`	A float. The variance of the outcome variable y. Must be strictly positive.
`sigma_x`	A matrix. The variance-covariance matrix of the covariates. Must be symetric positive-definite. Can also be a strictly positive float in the case of univariate regression.

Value

The joint log-likelihood of (y, X).

Details

Let C_j be the cluster of records corresponding to the j^{th} true entity, then assuming that there are N_{pop} true entities represented in the records (i.e. there are N_{pop} unique elements of `linkage`), `log_joint_lm_cluster` calculates

$$\sum_{j=1}^{N_{pop}} \log P\left([y, X]_{C_j} \mid \Lambda, \beta, \Sigma_{\widetilde{x}}, \Sigma_{x|\widetilde{x}}, \sigma^2_{y|\widetilde{x}}\right).$$

See Also

Other Cluster Likelihood Functions: `hitmiss()`, `log_distortion_cluster()`, `log_linkage_cluster()`, `log_lm_cluster()`, `recursive_cluster()`

log_linkage_cluster *Cluster log-likelihood.*

Description

Given a data.frame of records for a specific cluster, `log_recursive_cluster` loops over each key variable (each column of the cluster), calls `recursive_cluster`, and returns the sums of the log-likelihoods for each field.

Usage

```
log_linkage_cluster(cluster, thetas, alphas, key_vars)
```

Arguments

`cluster`	A data.frame. All of the records and their values for each key variable for a specific cluster.
`thetas`	A list of matrices. Each matrix corresponds to a key variable and contains two columns: 1) all of the values each key variable can take in the data, and 2) the corresponding empirical frequency of each value.
`alpha`	A data.frame. Contains a column `key_vars` with each of the key variables, and a column `alpha` with the corresponding distortion probabilities.

Value

The joint log-likelihood of the provided cluster.

Details

This function is used in the Metropolis-Hastings update step for the linkage structure Λ.

See Also

`recursive_cluster` for the actual recursion formula implemented for this function.

Other Cluster Likelihood Functions: `hitmiss()`, `log_distortion_cluster()`, `log_joint_lm_cluster()`, `log_lm_cluster()`, `recursive_cluster()`

log_lm_cluster *Conditional regression log-likelihood for single cluster.*

Usage

```
log_lm_cluster(cluster, B, Sigma)
```

Arguments

`cluster`	A data.fram. The model.frame of the regression corresponding to the records of a particular cluster. The first column must contain the outcome variable y and the remaining columns must contain the model covariates X.
`B`	A matrix. The first block component of the variance-covariance matrix.
`Sigma`	A matrix. The first block component of the variance-covariance matrix.

Value

The conditional log-likelihood of the given cluster.

Details

Let C_j be the cluster of records corresponding to the j^{th} true entity, then assuming that there are n records in C_j, and p covariates in X, `log_lm_cluster` calculates

$$\log P\left([y, X]_{C_j} \mid \beta, \Sigma_{\widetilde{x}}, \Sigma_{x|\widetilde{x}}, \sigma^2_{y|\widetilde{x}}\right) = \log N_{n(p+1)}\left(\mathbf{0},\ (\mathbf{1}_n\mathbf{1}_n') \otimes \mathbf{B} + \mathbf{I}_n \otimes \Sigma\right).$$

where $\mathbf{B} =$

$$\begin{vmatrix} \beta'\Sigma_{\widetilde{x}}\beta & \beta'\Sigma_{\widetilde{x}} \\ \Sigma_{\widetilde{x}}\beta & \Sigma_{\widetilde{x}} \end{vmatrix}$$

and $\Sigma =$

$$\begin{matrix} \sigma^2_{y|\widetilde{x}} & \mathbf{0} \\ \mathbf{0} & \Sigma_{x|\widetilde{x}}. \end{matrix}$$

See Also

Other Cluster Likelihood Functions: `hitmiss()`, `log_distortion_cluster()`, `log_joint_lm_cluster()`, `log_linkage_cluster()`, `recursive_cluster()`

`log_p_lambda` *Log prior probability of linkage struture*

Description

`log_p_lambda` calculates $\log P(\lambda_{ij} \mid \lambda_{-(ij)})$ where λ_{ij} is the cluster label of the j^{th} record in the i^{th} database.

Usage

```
log_p_lambda(lambda, record, linkage, lambda_prior)
```

Arguments

`lambda`	An integer. The cluster label corresponding to the record in the `record` row of the data.
`record`	An integer. The row corresponding to the current record being updated in the data.
`linkage`	An integer vector. The current cluster labels for each record.
`lambda_prior`	A list. A list containing the type of prior to use for the linkage structure and the corresponding parameters for the prior.

Value

$\log P(\lambda_{ij} \mid \lambda_{-(ij)})$

pyp_mean *Calculate the mean of a Pittman-Yor process.*

Description

Given N entities (in this case records), pyp_mean calculates the expected number of clusters for a Pittman-Yor process with parameters σ and ν,

$$\frac{\nu}{\sigma}\left[\frac{(\nu+\sigma)_{N\uparrow}}{\nu_{N\uparrow}}-1\right]$$

where $x_{s\uparrow}=\frac{\Gamma(x+s)}{\Gamma(x)}$.

Usage

```
pyp_mean(nu, sigma, N)
```

Arguments

nu, sigma	A float. PYP parameters such that $\sigma \in [0,1)$ and $\nu > -\sigma$ or $\sigma < 0$ and $\nu > m\|\sigma\|$ for some integer m.
N	An integer. The number of records.

Value

Mean of PYP prior.

See Also

Other PYP Functions: lgratio(), pyp_optim(), pyp_solve(), pyp_var(), pyp_warnings()

pyp_optim *Optimization function for backsolving parameters of a PYP prior.*

Description

Gives non-linear equations for ν and σ in terms of a known mean, variance, and N.

Usage

```
pyp_optim(par, mn, var, N)
```

Arguments

par	Parameters to optimize over.
mn	An integer. The mean condition of the PYP prior.
var	A float. The variance condition of the PYP prior. Must be strictly positive.
N	An integer. The number of records.

Value

A vector of residuals.

See Also

Other PYP Functions: `lgratio()`, `pyp_mean()`, `pyp_solve()`, `pyp_var()`, `pyp_warnings()`

`pyp_solve` *Solve for parameters a PYP prior given a mean, variance, and N.*

Description

`pyp_solve` finds the parameters ν and σ that give a PYP prior given the mean `mn`, variance `var`, for a given number of entities N.

Usage

```
pyp_solve(mn, var, N)
```

Arguments

`mn`	An integer. The mean condition of the PYP prior.
`var`	A float. The variance condition of the PYP prior. Must be strictly positive.
`N`	An integer. The number of records.

Value

A list. (σ, ν).

See Also

Other PYP Functions: `lgratio()`, `pyp_mean()`, `pyp_optim()`, `pyp_var()`, `pyp_warnings()`

Examples

```
pyp_solve(mn = 450, var = 500, N = 500)
```

pyp_var *Calculate the variance of a Pittman-Yor process.*

Description

Given N entities (in this case records), `pyp_var` calculates the variance of the number of clusters for a Pittman-Yor process with parameters σ and ν,

$$\frac{\nu(\nu+\sigma)}{\sigma^2}\frac{(\nu+2\sigma)_{N\uparrow}}{\nu_N\uparrow} - \left[\frac{\nu}{\sigma}\frac{(\nu+\sigma)_{N\uparrow}}{\nu_{N\uparrow}}\right]^2 - \frac{\nu}{\sigma}\frac{(\nu+\sigma)_{N\uparrow}}{\nu_{N\uparrow}}$$

where $x_{s\uparrow} = \frac{\Gamma(x+s)}{\Gamma(x)}$.

Usage

```
pyp_var(nu, sigma, N)
```

Arguments

`nu, sigma`	A float. The PYP parameters such that $\sigma \in [0, 1)$ and $\nu > -\sigma$ or $\sigma < 0$ and $\nu > m\|\sigma\|$ for some integer m.
`N`	An integer. The number of records.

Value

Variance of PYP prior.

See Also

Other PYP Functions: `lgratio()`, `pyp_mean()`, `pyp_optim()`, `pyp_solve()`, `pyp_warnings()`

pyp_warnings *PYP function parameter tests.*

Description

PYP function parameter tests.

Usage

```
pyp_warnings(nu, sigma, N)
```

Arguments

`nu, sigma`	A float. PYP parameters such that $\sigma \in [0, 1)$ and $\nu > -\sigma$ or $\sigma < 0$ and $\nu > m\|\sigma\|$ for some integer m.
`N`	An integer. The number of records.

Value

Warning if broken parameters are provided.

See Also

Other PYP Functions: `lgratio()`, `pyp_mean()`, `pyp_optim()`, `pyp_solve()`, `pyp_var()`

`recursiveRL` *Title*

Description

Title

Usage

```
recursiveRL(
  data,
  key_vars,
  sample_size,
  burnin = 0,
  thin = 1,
  n_chains = 1,
  progressbar = TRUE,
  alpha_init = NULL,
  alpha_prior,
  alpha_proposal = "RRW",
  alpha_drift = 0.05,
  lambda_init = NULL,
  lambda_prior
)
```

Arguments

`lambda_prior`

`recursive_cluster` *Cluster likelihood for the ℓth field.*

Description

Given the ℓ^{th} field of a cluster as a vector along with the empirical frequencies θ_ℓ and distortion probability, `recursive_cluster` uses a recursive formula to calculate the joint likelihood for the q^{th} cluster v_{C_q}.

Usage

```
recursive_cluster(cluster, thetas_l, alpha_l)
```

Arguments

`cluster`	A vector. The vector of strings/factors for the ℓ^{th} field of the selected cluster.
`thetas_l`	A matrix. The first column of `thetas_l` must be the possible values that `cluster`, and the second column should contain the corresponding frequencies of each value in the whole set of records.
`alpha_l`	A float. The distortion probability of the ℓ^{th} field.

Value

The joint likelihood of the ℓ^{th} field of a cluster.

Details

This function is used in the Metropolis-Hastings update step for the linkage structure Λ.

The recursive formula for $P(v_{C_q,\ell} \mid \alpha_\ell, \lambda)$ is given by

$$P(v_{C_q,\ell} \mid \alpha_\ell, \lambda) = \theta_{v_{ij\ell}},$$

if $v_{ij\ell} = v_{C_q,\ell}$, and

$P(v_{C_q,\ell} \mid \alpha_\ell, \lambda) =$

$$(1-\alpha_\ell)\left[\prod_{(i',j')\in C_{q\setminus(ij)}} (1-\alpha_\ell)\delta(v_{i'j'\ell}, v_{ij\ell}) + \alpha_\ell\theta_{v_{i'j'},\ell}\right]\theta_{v_{ij\ell}} + \alpha_\ell\theta_{v_{ij\ell}}P(V_{C_q\setminus(ij),\ell} \mid \alpha_\ell, \lambda)$$

otherwise.

See Also

Other Cluster Likelihood Functions: `hitmiss()`, `log_distortion_cluster()`, `log_joint_lm_cluster()`, `log_linkage_cluster()`, `log_lm_cluster()`

Index