Package 'BayesMallowsSMC2'

February 4, 2026

Type Package

Title Nested Sequential Monte Carlo for the Bayesian Mallows Model

Version 0.2.1

Description Provides nested sequential Monte Carlo algorithms for performing sequential inference in the Bayesian Mallows model, which is a widely used probability model for rank and preference data. The package implements the SMC2 (Sequential Monte Carlo Squared) algorithm for handling sequentially arriving rankings and pairwise preferences, including support for complete rankings, partial rankings, and pairwise comparisons. The methods are based on Sorensen (2025) <doi:10.1214/25-BA1564>.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp, ggplot2, Rdpack

Depends R (>= 4.1.0)

Suggests testthat (>= 3.0.0), label.switching (>= 1.8)

RdMacros Rdpack

Config/testthat/edition 3

NeedsCompilation yes

Author Oystein Sorensen [aut, cre] (ORCID: <https://orcid.org/0000-0003-0724-3542>)

Maintainer Oystein Sorensen <oystein.sorensen.1985@gmail.com>

Repository CRAN

Date/Publication 2026-02-04 15:50:10 UTC

Contents

`complete_rankings` *Simulated Data*

Description

Simulated Data

Usage

```
complete_rankings
```

Format

`complete_rankings`**:**

A data frame with 100 rows and 7 columns:

timepoint Timepoint

user User id

item1, item2, item3, item4, item5 Ranking given to the item.

compute_sequentially *Compute the Bayesian Mallows model sequentially*

Description

Fits the Bayesian Mallows model to sequentially arriving ranking or preference data using the SMC2 (Sequential Monte Carlo Squared) algorithm. This function can handle complete rankings, partial rankings, and pairwise preference data, and supports mixture models with multiple clusters.

Usage

```
compute_sequentially(
  data,
  hyperparameters = set_hyperparameters(),
  smc_options = set_smc_options(),
  topological_sorts = NULL
)
```

Arguments

`data` A dataframe containing partial rankings or pairwise preferences. If `data` contains complete or partial rankings, it must have the following columns:

- `timepoint`: a numeric vector denoting the timepoint, starting at 1.
- `user`: a vector identifying the user.
- `item1`: ranking of item 1.
- `item2`: ranking of item 2.
- etc.

If data contains pairwise preferences, it must have the following structure:

- `timepoint`: a numeric vector denoting the timepoint, starting at 1.
- `user`: a vector identifying the user.
- `top_item`: identifier for the preferred item.
- `bottom_item`: identifier for the dispreferred item.

`hyperparameters` A list returned from `set_hyperparameters()`.

`smc_options` A list returned from `set_smc_options()`

`topological_sorts` A list returned from `precompute_topological_sorts()`. Only used with preference data, and defaults to `NULL`.

Details

The function implements the SMC2 algorithm for sequential Bayesian inference in the Mallows model. At each timepoint, it updates the particle approximation to the posterior distribution as new

ranking or preference data arrives. The algorithm automatically performs resampling and rejuvenation steps when the effective sample size drops below the specified threshold.

The returned object has S3 methods for printing (print.BayesMallowsSMC2), summarizing (summary.BayesMallowsSMC2), and plotting (plot.BayesMallowsSMC2). For visualization of parameter evolution over time, see `trace_plot()`.

Value

An object of class `BayesMallowsSMC2`, which is a list containing:

alpha A matrix of dispersion parameter values with dimensions `[n_clusters, n_particles]`. Each column represents one particle, and each row represents one cluster.

rho A 3-dimensional array of latent ranking values with dimensions `[n_items, n_clusters, n_particles]`. Entry `[i, k, j]` gives the rank of item `i` in cluster `k` for particle `j`.

tau A matrix of precision parameter values (for mixture models) with dimensions `[n_clusters, n_particles]`. Each column represents one particle.

cluster_probabilities A 3-dimensional array of cluster assignment probabilities (for mixture models) with dimensions `[n_particles, n_users, n_clusters]`. Only present when `n_clusters` > 1.

importance_weights A numeric vector of length `n_particles` containing the normalized importance weights for each particle.

ESS A numeric vector of length `n_timepoints` containing the effective sample size at each timepoint.

resampling An integer vector of length `n_timepoints` indicating whether resampling occurred at each timepoint (1 = yes, 0 = no).

n_particle_filters An integer vector of length `n_timepoints` showing the number of particle filters used at each timepoint.

log_marginal_likelihood A numeric value giving the estimated log marginal likelihood of the data.

alpha_traces A list of parameter traces (only if `trace = TRUE` in `set_smc_options()`). Each element contains alpha values at one timepoint.

tau_traces A list of parameter traces (only if `trace = TRUE` in `set_smc_options()`). Each element contains tau values at one timepoint.

rho_traces A list of parameter traces (only if `trace = TRUE` in `set_smc_options()`). Each element contains rho values at one timepoint.

log_importance_weights_traces A list of importance weight traces (only if `trace = TRUE` in `set_smc_options()`).

latent_rankings_traces A list of latent ranking traces (only if `trace_latent = TRUE` in `set_smc_options()`).

References

Sørensen Ø, Stein A, Netto WL, Leslie DS (2025). "Sequential Rank and Preference Learning with the Bayesian Mallows Model." *Bayesian Analysis*, 1 – 26. doi:10.1214/25BA1564.

Examples

```
# Compute the model sequentially with complete rankings
mod <- compute_sequentially(
  complete_rankings,
  hyperparameters = set_hyperparameters(n_items = 5),
  smc_options = set_smc_options(n_particles = 100, n_particle_filters = 1)
)

# Print the model summary
mod

# Plot posterior distribution of alpha
plot(mod, parameter = "alpha")
```

---

mixtures *Simulated Two-Component Mixtures Data*

Description

Simulated Two-Component Mixtures Data

Usage

```
mixtures
```

Format

`mixtures`**:**

A data frame with 400 rows and 7 columns:

timepoint Timepoint

user User id

item1, item2, item3, item4, item5 Ranking given to the item.

---

pairwise_preferences *Simulated Data with Pairwise Preferences*

Description

Simulated Data with Pairwise Preferences

Usage

```
pairwise_preferences
```

Format

`pairwise_preferences`**:**

A data frame with 400 rows and 4 columns:

timepoint Timepoint

user User id

top_item Item preferred in the comparison.

bottom_item Item disfavored in the comparison.

---

partial_rankings *Simulated Data with Missing Values*

Description

Simulated Data with Missing Values

Usage

```
partial_rankings
```

Format

`partial_rankings`**:**

A data frame with 100 rows and 7 columns:

timepoint Timepoint

user User id

item1, item2, item3, item4, item5 Ranking given to the item.

---

plot.BayesMallowsSMC2 *Plot Posterior Distributions for BayesMallowsSMC2 Objects*

Description

Visualize posterior distributions of the parameters of the Bayesian Mallows model after the final timepoint.

Usage

```
## S3 method for class 'BayesMallowsSMC2'
plot(x, parameter = "alpha", items = NULL, ...)
```

Arguments

x	An object of class BayesMallowsSMC2, returned from compute_sequentially().
parameter	Character string defining the parameter to plot. Available options are "alpha" (default), "rho", and "tau".
items	The items to study in the plot for rho. Either a vector of item names or a vector of indices (1 to n_items). If NULL and there are more than 5 items, 5 items are selected randomly. Only used when parameter = "rho".
...	Other arguments (currently unused).

Details

The function uses importance weights from the SMC algorithm to compute weighted posterior distributions. For mixture models (multiple clusters), separate plots are created for each cluster using faceting.

For parameter = "rho", if the number of items is large (> 5) and items is not specified, a random subset of 5 items is selected for visualization. The random selection respects the current random seed, so results are reproducible if you set a seed before calling the function (e.g., with set.seed()).

Value

A ggplot object showing:

- For parameter = "alpha": A histogram of the posterior distribution of the dispersion parameter.
- For parameter = "tau": A histogram of the posterior distribution of the precision parameter.
- For parameter = "rho": Bar charts showing the posterior probability of each ranking for each item separately.

Examples

```
# Fit a model with complete rankings
mod <- compute_sequentially(
  complete_rankings,
  hyperparameters = set_hyperparameters(n_items = 5),
  smc_options = set_smc_options(n_particles = 100, n_particle_filters = 1)
)

# Plot alpha (default)
plot(mod)

# Plot tau
plot(mod, parameter = "tau")

# Plot rho for specific items
plot(mod, parameter = "rho", items = c(1, 3, 5))

# Plot rho for all items (if <= 5 items)
plot(mod, parameter = "rho")
```

precompute_topological_sorts
Precompute All Topological Sorts

Description

This function precomputes all topological sorts for a given preference matrix. Topological sorts are consistent orderings of items that respect the given pairwise preference constraints.

Usage

```
precompute_topological_sorts(prefs, n_items, save_frac)
```

Arguments

prefs	A matrix representing the preference relations. This matrix must have two columns, the first of which represents the preferred item and the second of which represents the disfavored item.
n_items	An integer specifying the number of items to sort.
save_frac	Number between 0 and 1 specifying which fraction of sorts to save.

Details

The function generates all possible topological sorts for the provided preference matrix and saves approximately save_frac of the sorts in a matrix which is returned.

Value

A list with two elements:

sort_count An integer giving the total number of topological sorts.

sort_matrix A matrix where each column represents one topological sort. The number of columns is approximately save_frac times sort_count. If save_frac = 0, this is an empty matrix with dimensions c(0, 0).

Examples

```
# Extract preferences from user 1 in the included example data.
prefs <- pairwise_preferences[
 pairwise_preferences$user == 1, c("top_item", "bottom_item"), drop = FALSE]

# Generate all topological sorts, but don't save them:
sorts <- precompute_topological_sorts(
  prefs = as.matrix(prefs),
  n_items = 5,
  save_frac = 0
```

```
)
# Number of sorts
sorts$sort_count
# Empty matrix
sorts$sort_matrix

# Generate all topological sorts and save them:
sorts <- precompute_topological_sorts(
  prefs = as.matrix(prefs),
  n_items = 5,
  save_frac = 1
)
# Number of sorts
sorts$sort_count
# Matrix with all of them
sorts$sort_matrix
```

`print.BayesMallowsSMC2` *Print Method for BayesMallowsSMC2 Objects*

Description

Prints a summary of a BayesMallowsSMC2 object returned by `compute_sequentially()`.

Usage

```
## S3 method for class 'BayesMallowsSMC2'
print(x, ...)
```

Arguments

`x`	An object of class `BayesMallowsSMC2`.
`...`	Additional arguments (currently unused).

Details

The print method displays key information about the fitted Bayesian Mallows model, including:

- Number of particles
- Number of timepoints
- Number of items
- Number of clusters
- Log marginal likelihood
- Final effective sample size (ESS)
- Number of resampling events

Value

Invisibly returns the input object `x`.

Examples

```
# Fit a model with complete rankings
set.seed(123)
mod <- compute_sequentially(
  complete_rankings,
  hyperparameters = set_hyperparameters(n_items = 5),
  smc_options = set_smc_options(n_particles = 100, n_particle_filters = 1)
)

# Print the model
print(mod)

# or simply
mod
```

---

print.summary.BayesMallowsSMC2 *Print Method for summary.BayesMallowsSMC2 Objects*

Description

Prints a summary of a BayesMallowsSMC2 model.

Usage

```
## S3 method for class 'summary.BayesMallowsSMC2'
print(x, ...)
```

Arguments

`x`	An object of class `summary.BayesMallowsSMC2`.
`...`	Additional arguments (currently unused).

Value

Invisibly returns the input object `x`.

`set_hyperparameters` *Set hyperparameters*

Description

Set the hyperparameters for the Bayesian Mallows model. This function creates a list of hyperparameter values that can be passed to `compute_sequentially()`.

Usage

```
set_hyperparameters(
  n_items,
  alpha_shape = 1,
  alpha_rate = 0.5,
  cluster_concentration = 10,
  n_clusters = 1
)
```

Arguments

`n_items`	Integer defining the number of items.
`alpha_shape`	Shape parameter of the gamma prior distribution for the scale parameter alpha. Defaults to 1.
`alpha_rate`	Rate parameter of the gamma prior distribution for the scale parameter alpha. Defaults to 0.5.
`cluster_concentration`	Concentration parameter of the Dirichlet distribution for cluster probabilities. Only used when `n_clusters` > 1. Defaults to 10.
`n_clusters`	Integer defining the number of clusters. Defaults to 1.

Value

A list with components `n_items`, `alpha_shape`, `alpha_rate`, `cluster_concentration`, and `n_clusters`.

Examples

```
# Example: Set hyperparameters and use them with partial rankings
# Set hyperparameters with default values
hyperparams1 <- set_hyperparameters(n_items = 5)

# Set hyperparameters with custom prior for alpha
# A larger alpha_shape and smaller alpha_rate increases the prior mean
hyperparams2 <- set_hyperparameters(
  n_items = 5,
  alpha_shape = 2,
  alpha_rate = 1
)
```

```
# Use the hyperparameters with compute_sequentially
# This example uses partial rankings with a small number of particles
# for fast execution suitable for CRAN checks
set.seed(123)
mod <- compute_sequentially(
  partial_rankings,
  hyperparameters = hyperparams2,
  smc_options = set_smc_options(
    n_particles = 20,
    n_particle_filters = 4,
    max_rejuvenation_steps = 3
  )
)
```

`set_smc_options` *Set SMC options*

Description

Configure the SMC2 (Sequential Monte Carlo Squared) algorithm for the Bayesian Mallows model. This function sets parameters that control the particle filter, resampling strategy, and diagnostic output.

Usage

```
set_smc_options(
  n_particles = 1000,
  n_particle_filters = 50,
  max_particle_filters = 10000,
  resampling_threshold = n_particles/2,
  doubling_threshold = 0.2,
  max_rejuvenation_steps = 20,
  metric = "footrule",
  resampler = "multinomial",
  latent_rank_proposal = "uniform",
  verbose = FALSE,
  trace = FALSE,
  trace_latent = FALSE
)
```

Arguments

`n_particles`	Integer specifying the number of particles to use in the outer SMC loop. More particles generally improve approximation accuracy but increase computational cost. Defaults to 1000.

`n_particle_filters`
: Integer specifying the initial number of particle filters for each particle in the inner loop. This controls the granularity of the latent rank estimation. Defaults to 50.

`max_particle_filters`
: Integer specifying the maximum number of particle filters allowed. The algorithm can adaptively increase the number of filters up to this limit when the acceptance rate is low. Defaults to 10000.

`resampling_threshold`
: Numeric specifying the effective sample size threshold for triggering resampling. When the effective sample size falls below this threshold, the particles are resampled to avoid degeneracy. Defaults to `n_particles / 2`.

`doubling_threshold`
: Numeric threshold for particle filter doubling. If the acceptance rate of the rejuvenation step falls below this threshold, the number of particle filters is doubled (up to `max_particle_filters`) to improve mixing. Should be between 0 and 1. Defaults to 0.2.

`max_rejuvenation_steps`
: Integer specifying the maximum number of rejuvenation MCMC steps to perform. The rejuvenation step helps maintain particle diversity. The algorithm stops early if the number of unique particles exceeds half the total number of particles. Defaults to 20.

`metric`
: Character string specifying the distance metric to use for comparing rankings. Options are `"footrule"` (default), `"spearman"`, `"kendall"`, `"cayley"`, `"hamming"`, or `"ulam"`. The choice of metric affects the likelihood function in the Mallows model.

`resampler`
: Character string specifying the resampling algorithm. Options are `"multinomial"` (default), `"residual"`, `"stratified"`, or `"systematic"`. Different resamplers have different variance properties.

`latent_rank_proposal`
: Character string specifying the proposal distribution for latent ranks in the Metropolis-Hastings step. Options are `"uniform"` (default) or `"pseudo"`. The `"pseudo"` option can provide better proposals for partial rankings.

`verbose`
: Logical indicating whether to print progress messages during computation. Defaults to `FALSE`.

`trace`
: Logical specifying whether to save static parameters (alpha, rho, cluster probabilities) at each timestep. This is useful for diagnostic purposes but increases memory usage. Defaults to `FALSE`.

`trace_latent`
: Logical specifying whether to sample and save one complete set of latent rankings for each particle at each timepoint. This can be used to inspect the evolution of rankings over time but substantially increases memory usage. Defaults to `FALSE`.

Details

The SMC2 algorithm uses a nested particle filter structure:

- The outer loop maintains `n_particles` particles, each representing a hypothesis about the static parameters (alpha, rho, cluster probabilities).
- The inner loop uses `n_particle_filters` particle filters per outer particle to estimate the latent rankings.
- When new data arrives, the algorithm updates the particle weights and performs rejuvenation MCMC steps to maintain diversity.
- If particle degeneracy occurs (effective sample size below `resampling_threshold`), particles are resampled.
- If the acceptance rate during rejuvenation is low (below `doubling_threshold`), the number of particle filters is adaptively doubled.

For computational efficiency with CRAN examples, use smaller values such as `n_particles = 100` and `n_particle_filters = 1`.

Value

A list containing all the specified options, suitable for passing to `compute_sequentially()`.

See Also

`compute_sequentially()`, `set_hyperparameters()`

Examples

```
# Basic usage with default settings
opts <- set_smc_options()

# Customize for faster computation (suitable for CRAN examples)
opts_fast <- set_smc_options(
  n_particles = 100,
  n_particle_filters = 1
)

# Use with complete rankings data
mod <- compute_sequentially(
  complete_rankings,
  hyperparameters = set_hyperparameters(n_items = 5),
  smc_options = set_smc_options(n_particles = 100, n_particle_filters = 1)
)

# Customize resampling and metric
opts_custom <- set_smc_options(
  n_particles = 200,
  n_particle_filters = 10,
  resampler = "stratified",
  metric = "kendall"
)

# Enable diagnostic output
opts_trace <- set_smc_options(
  n_particles = 100,
```

```
  n_particle_filters = 1,
  verbose = TRUE,
  trace = TRUE
)

# For partial rankings with more rejuvenation steps
mod_partial <- compute_sequentially(
  partial_rankings[1:10, ],
  hyperparameters = set_hyperparameters(n_items = 5),
  smc_options = set_smc_options(
    n_particles = 30,
    n_particle_filters = 5,
    max_rejuvenation_steps = 10,
    latent_rank_proposal = "uniform"
  )
)
```

`summary.BayesMallowsSMC2` *Summary Method for BayesMallowsSMC2 Objects*

Description

Creates a summary of a BayesMallowsSMC2 object returned by `compute_sequentially()`.

Usage

```
## S3 method for class 'BayesMallowsSMC2'
summary(object, ...)
```

Arguments

`object`	An object of class `BayesMallowsSMC2`.
`...`	Additional arguments (currently unused).

Details

The summary method creates a summary object that includes:

- Number of particles
- Number of timepoints
- Number of items
- Number of clusters
- Log marginal likelihood
- Final effective sample size (ESS)
- Number of resampling events
- Posterior mean of alpha for each cluster
- Posterior standard deviation of alpha for each cluster

Value

An object of class `summary.BayesMallowsSMC2`, which is a list containing summary information about the model.

Examples

```
# Fit a model with complete rankings
set.seed(123)
mod <- compute_sequentially(
  complete_rankings,
  hyperparameters = set_hyperparameters(n_items = 5),
  smc_options = set_smc_options(n_particles = 100, n_particle_filters = 1)
)

# Create summary
summary(mod)
```

---

trace_plot *Create Trace Plots for BayesMallowsSMC2 Objects*

Description

Visualize the timeseries dynamics of the alpha and tau parameters across timepoints. This function creates trace plots similar to Figure 4 (left) in Sørensen Ø, Stein A, Netto WL, Leslie DS (2025). "Sequential Rank and Preference Learning with the Bayesian Mallows Model." *Bayesian Analysis*, 1 – 26. doi:10.1214/25BA1564..

Usage

```
trace_plot(x, parameter = "alpha", ...)
```

Arguments

`x`	An object of class `BayesMallowsSMC2`, returned from `compute_sequentially()` with `trace = TRUE` in `set_smc_options()`.
`parameter`	Character string defining the parameter to plot. Available options are `"alpha"` (default) and `"tau"`.
`...`	Other arguments (currently unused).

Details

This function requires that the model was fitted with `trace = TRUE` in the `smc_options`. The trace contains the parameter values at each timepoint, which allows visualization of how the posterior distribution evolves as more data arrives sequentially.

For mixture models (multiple clusters), separate trace plots are created for each cluster using faceting.

The shaded area represents the 95% credible interval (from 2.5% to 97.5% quantiles) of the posterior distribution at each timepoint, computed using the importance weights from the SMC algorithm.

Value

A ggplot object showing the evolution of the parameter over time. For each timepoint, the plot shows:

- The weighted mean (solid line)
- The weighted 0.025 and 0.975 quantiles (shaded area representing the 95% credible interval)

References

Sørensen Ø, Stein A, Netto WL, Leslie DS (2025). "Sequential Rank and Preference Learning with the Bayesian Mallows Model." *Bayesian Analysis*, 1 – 26. doi:10.1214/25BA1564.

Examples

```
# Fit a model with trace enabled
mod <- compute_sequentially(
  complete_rankings,
  hyperparameters = set_hyperparameters(n_items = 5),
  smc_options = set_smc_options(
    n_particles = 100,
    n_particle_filters = 1,
    trace = TRUE
  )
)

# Create trace plot for alpha (default)
trace_plot(mod)
```

Index