Package 'bayesWatch'

May 7, 2026

Type Package

Title Bayesian Change-Point Detection for Process Monitoring with Fault Detection

Version 0.1.4

Maintainer Alexander C. Murph <murph290@gmail.com>

Description Bayes Watch fits an array of Gaussian Graphical Mixture Models to groupings of homogeneous data in time, called regimes, which are modeled as the observed states of a Markov process with unknown transition probabilities. In doing so, Bayes Watch defines a posterior distribution on a vector of regime assignments, which gives meaningful expressions on the probability of every possible change-point. Bayes Watch also allows for an effective and efficient fault detection system that assesses what features in the data where the most responsible for a given change-point. For further details, see: Alexander C. Murph et al. (2023) <doi:10.48550/arXiv.2310.02940>.

Copyright file COPYRIGHTS

License GPL-3

Imports Rcpp (>= 1.0.7), parallel (>= 3.6.2), Matrix, Hotelling, CholWishart, ggplot2, gridExtra (>= 0.9.1), BDgraph, methods, MASS, stats, ess

LinkingTo Rcpp, RcppArmadillo, RcppEigen, Matrix, CholWishart, BH

Depends R (>= 3.5.0)

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation yes

Repository CRAN

Date/Publication 2025-08-30 20:20:02 UTC

Author Alexander C. Murph [aut, cre],
Reza Mohammadi [ctb, cph],
Alex Lenkoski [ctb, cph],
Andrew Johnson [ctb]

Contents

bayeswatch 2
detect_faults 5
full_data 6
get_point_estimate 6
plot.bayesWatch 7
print.bayesWatch 7

Index **8**

`bayeswatch` *Fit a bayesWatch object.*

Description

Main method of package. MCMC sampling for change-point probabilities with fault detection according to the model by Murph et al. 2023. Creates a bayesWatch object for analysis of change-points.

Usage

```
bayeswatch(
  data_woTimeValues,
  time_of_observations,
  time_points,
  variable_names = 1:ncol(data_woTimeValues),
  not.cont = NULL,
  iterations = 100,
  burnin = floor(iterations/2),
  lower_bounds = NULL,
  upper_bounds = NULL,
  ordinal_indicators = NULL,
  list_of_ordinal_levels = NULL,
  categorical_indicators = NULL,
  previous_states = NULL,
  previous_model_fits = NULL,
  linger_parameter = 500,
  move_parameter = 100,
  g.prior = 0.2,
  set_G = NULL,
  wishart_df_initial = 1500,
  lambda = 1500,
  g_sampling_distribution = NULL,
  n.cores = 1,
  scaleMatrix = NULL,
  allow_for_mixture_models = FALSE,
  dirichlet_prior = 0.001,
```

```
  component_truncation = 7,
  regime_truncation = 15,
  hyperprior_b = 20,
  model_params_save_every = 5,
  simulation_iter = NULL,
  T2_window_size = 3,
  determining_p_cutoff = FALSE,
  prob_cutoff = 0.5,
  model_log_type = "NoModelSpecified",
  regime_selection_multiplicative_prior = 2,
  split_selection_multiplicative_prior = 2,
  is_initial_fit = TRUE,
  verbose = FALSE
)
```

Arguments

`data_woTimeValues`	matrix. Raw data matrix without datetime stamps.
`time_of_observations`	vector. Datetime stamps for every data instance in data_woTimeValues.
`time_points`	vector. Time points that mark each 'day' of time. Range should include every datetime in time_of_observations.
`variable_names`	vector. Vector of names of columnsof data_woTimeValues.
`not.cont`	vector. Indicator variable as to which columns are discrete.
`iterations`	integer. Number of MCMC samples to take (including burn-in).
`burnin`	integer. Number of burn-in samples. iterations > burnin necessarily.
`lower_bounds`	vector. Lower bounds for each data column.
`upper_bounds`	vector. Upper bounds for each data column.
`ordinal_indicators`	vector. Discrete values, one for each column, indicating which variables are ordinal.
`list_of_ordinal_levels`	vector. Discrete values, one for each column, indicating which variables are part of the same ordinal group.
`categorical_indicators`	vector. Each nominal d categorical variable must be broken down into d different indicator variables. This vector marks which variables are such indicators.
`previous_states`	vector. Starting regime vector, if known, of the same length as the number of 'days' in time_points.
`previous_model_fits`	rlist. Starting parameter fits corresponding to regime vector previous_states.
`linger_parameter`	float. Prior parameter for Markov chain probability matrix. Larger = less likely to change states.

`move_parameter` float. Prior parameter for Markov chain probability matrix. Larger = more likely to change states.

`g.prior` float in (0,1). Prior probability on edge inclusion for graph structure G.

`set_G` matrix. Starting graph structure, if known.

`wishart_df_initial` integer (>= 3). Starting DF for G-Wishart prior.

`lambda` float. Parameter for NI-G-W prior, controls affect of precision sample on the center sample.

`g_sampling_distribution` matrix. Prior probability on edge inclusion if not uniform across G.

`n.cores` integer. Number of cores available for parallelization.

`scaleMatrix` matrix. Parameter for NI-G-W prior.

`allow_for_mixture_models` logical. Whether or not method should fix mixture distributions to regimes.

`dirichlet_prior` float. Parameter for the dirichlet process for fitting components in the mixture model.

`component_truncation` integer. Maximum component allowed. Should be sufficiently large.

`regime_truncation` integer. Maximum regime allowed. Should be sufficiently large.

`hyperprior_b` integer. Hyperprior on Wishart distribution fit to the scaleMatrix.

`model_params_save_every` integer. How frequently to save model fits for the fault detection method.

`simulation_iter` integer. Used for simulation studies. Deprecated value at package launch.

`T2_window_size` integer. Length of sliding window for Hotelling T2 pre-step. Used when an initial value for previous_states is not provided.

`determining_p_cutoff` logical. Method for estimating the probability cutoff on the posterior distribution for determining change-points. Deprecated at package launch date.

`prob_cutoff` float. Changepoints are determined (for fault detection process) if posterior probability exceeds this value.

`model_log_type` character vector. The type of log (used to distinguish logfiles).

`regime_selection_multiplicative_prior` float. Must be >=1. Gives additional probability to the most recent day for the selection of a new split point.

`split_selection_multiplicative_prior` float.

`is_initial_fit` logical. True when there is no previously fit bayesWatch object fed through the algorithm..

`verbose` logical. Prints verbose model output for debugging when TRUE. It is highly recommended that you pipe this to a text file.

Value

bayesWatch object. A model fit for the analysis of posterior change-points and fault detection.

Examples

```
data("full_data", package = "bayesWatch")
head(full_data)
data("day_of_observations", package = "bayesWatch")
head(day_of_observations)
data("day_dts", package = "bayesWatch")
head(day_dts)


library(bayesWatch)
data("full_data")
data("day_of_observations")
data("day_dts")

x       = bayeswatch(full_data, day_of_observations, day_dts,
                     iterations = 500, g.prior = 1, linger_parameter = 20, n.cores=3,
                     wishart_df_initial = 3, hyperprior_b = 3, lambda = 5)

print(x)
plot(x)
detect_faults(x)
```

`detect_faults` *Determine the cause of a change-point.*

Description

Prints out fault detection graphics given a bayesWatch object. This method can only be run if fault detection was run on the bayesWatch fit (if model_params_save_every < iterations).

Usage

```
detect_faults(regime_fit_object)
```

Arguments

`regime_fit_object`
: bayesWatch object. Fit with main method of package.

Value

ggplot object. Fault detection graphs.

full_data *Simulated Data with Imposed Change-points.*

Description

Data simulated using the BDgraph package. A change-point is imposed between days 5 and 6. The change only occurs in variables 3 and 4.

Format

'full_data' is a matrix, the latter two are vectors.

Details

'full_data' is a data frame with 1,000 rows and 5 columns. 'day_of_observations'; is a timestamp of each of 'full_data''s 1,000 rows. 'day_dts'; is a vector of unique elements from 'day_of_observations'..

Examples

```
data("full_data", package = "bayesWatch")
head(full_data)
data("day_of_observations", package = "bayesWatch")
head(day_of_observations)
data("day_dts", package = "bayesWatch")
head(day_dts)
```

get_point_estimate *Create an estimate on posterior distribution of change-points.*

Description

Given a bayesWatch object and a probability cutoff, finds change-points.

Usage

```
get_point_estimate(regime_fit_object, prob_cutoff)
```

Arguments

regime_fit_object	bayesWatch object. Fit with the bayesWatch method.
prob_cutoff	float in (0,1). Posterior probabilities above this cutoff will be considered change-points.

Value

vector. Indicator values corresponding to change-point locations.

`plot.bayesWatch` *Print function for a bayesWatch object. Prints only the posterior change-point probabilities.*

Description

Print function for a bayesWatch object. Prints only the posterior change-point probabilities.

Arguments

`x`	bayesWatch object. Fit from bayesWatch main method.
`...`	Additional plotting arguments.

`print.bayesWatch` *Print function for a bayesWatch object. Prints only the posterior change-point probabilities.*

Description

Print function for a bayesWatch object. Prints only the posterior change-point probabilities.

Arguments

`x`	bayesWatch object. Fit from bayesWatch main method.
`...`	Additional plotting arguments.

Index

∗ **datasets**
full_data, 6

bayeswatch, 2

day_dts *(*full_data*)*, 6
day_of_observations *(*full_data*)*, 6
detect_faults, 5

full_data, 6

get_point_estimate, 6

plot.bayesWatch, 7
print.bayesWatch, 7