Package: bulletcp (via r-universe)

September 13, 2024

Type Package

Title Automatic Groove Identification via Bayesian Changepoint Detection

Version 1.0.0

Maintainer Nathaniel Garton <nate.garton13@gmail.com>

Description Provides functionality to automatically detect groove locations via a Bayesian changepoint detection method to be used in the data preprocessing step of forensic bullet matching algorithms. The methods in this package are based on those in Stephens (1994) <doi:10.2307/2986119>. Bayesian changepoint detection will simply be an option in the function from the package 'bulletxtrctr' which identifies the groove locations.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

Depends R (>= 2.10), mvtnorm, dplyr, stats, assertthat

Suggests knitr, rmarkdown, ggplot2

VignetteBuilder knitr

Imports Rdpack

RdMacros Rdpack

Repository https://nategarton13.r-universe.dev

RemoteUrl https://github.com/nategarton13/bulletcp

RemoteRef HEAD

RemoteSha 19b53e3139331c7f09a6091fab2f445822b3e21d

Contents

detect_cp 2
example_data 4

get_grooves_bcp 5
imputeGP 5
mlgp 6
robust_loess_fit 8
runmcmc_cp0 8
runmcmc_cp1 10
runmcmc_cp1_left 12
runmcmc_cp1_right 14
runmcmc_cp2 16
runmcmc_cpall 18

Index **22**

`detect_cp` *Impute data and estimate groove locations.*

Description

This function is mostly just a wrapper function which calls the functions necessary to impute missing data, run the changepoint Gibbs algorithms, and select MAP estimates of the changepoint locations. Much less output is given for this function than for the functions called by this function. If all goes well, one should only need to explicitly use this function to estimate groove locations. Note that because this function calls the functions which do the Gibbs sampling, all of the input required for those functions is required by this function.

Usage

```
detect_cp(
  data,
  iter = 5000,
  start.vals = NA,
  prop_var = NA,
  cp_prop_var = NA,
  tol_edge = 50,
  tol_cp = 1000,
  warmup = 200,
  verbose = FALSE,
  prior_numcp = rep(1/4, times = 4),
  est_impute_par = FALSE,
  impute_par = c(0.8, 15)
)
```

Arguments

`data`	Data frame with columns "x" and "y." "x" is a column of the locations of the observed residual values, y.
`iter`	Number of iterations after warmup.

`start.vals` Starting values for the changepoint algorithm. Either NA valued or a named list of lists. If list, the names of the lists should be "cp2","cp1", and "cp0". Each list posessing one of those aforementioned names is a list of starting values identical to what would be given if the changepoint algorithm were to be run with the corresponding number of specified changepoints. List with elements "sigma", "l", "cp", "beta", and "intercept." "sigma" and "l" are 3 element vectors where the first element is for the data on the left groove. The second element is for the land engraved area, and the third element is for the right groove. "cp" is the vector of changepoint starting values. "beta" and "intercept" are two element vectors of the slope and intercept for the left and right groove engraved area respectively. If NA, default starting values will be used. Note that the changepoint starting values should always be near the edges of the data.

`prop_var` Either NA valued or a list of named lists. If list, the names of the lists should be "cp2","cp1", and "cp0". Each list posessing one of those aforementioned names is a list of proposal covariance matrices identical to what would be given if the changepoint algorithm were to be run with the corresponding number of specified changepoints.

`cp_prop_var` The proposal variance-covariance matrix for the changepoints. Can either be NA or a named list. If list, the names of the list items should be "cp2", "cp1" where each is the appropriate proposal variance/covariance matrix for the number of changepoints.

`tol_edge` This parameter controls how close changepoint proposals can be to the edge of the data before getting automatically rejected. For example, a value of 10 means that the changepoint will be automatically rejected if either of the proposal changepoints is within a distance of 10 x-values from either edge.

`tol_cp` This parameter controls how close changepoint proposals can be to each other before getting automatically rejected. For example, a value of 10 means that the changepoint will be automatically rejected if either of the proposal changepoints is within a distance of 10 x-values from either each other.

`warmup` The number of warmup iterations. This should be set to a very small number of iterations, as using too many iterations as warmup risks moving past the changepoints and getting stuck in a local mode. Default is set to 500.

`verbose` Logical value indicating whether to print the iteration number and the parameter proposals.

`prior_numcp` This is a vector with four elements giving the prior probabilities for the zero changepoint model, the one changepoint on the left model, the one changepoint on the right model, and the two changepoint model, in that order. Note that, practically, because the likelihood values are so large, only very strong priors will influence the results.

`est_impute_par` Logical value indicating whether parameters for the Gaussian process imputation should be estimated before actually doing the imputation. Default is FALSE, in which case the default imputation standard deviation is 0.8 and the length scale is 15. The covariance function is a squared exponential. These values have worked well in testing.

`impute_par` A two element vector containing the standard deviation and length scale (in that order) to use for the Gaussian process imputation. These values will not be used if the est_impute_par argument is set to TRUE.

Value

A named list containing the output from variable_cp_gibbs function, the range of data that was actually used for the changepoint algorithm (since it doesn't impute values past the outermost non-missing values), and the estimated groove locations.

Examples

```
# Fake data
sim_groove <- function(beta = c(-0.28,0.28), a = 125)
{
    x <- seq(from = 0, to = 2158, by = 20)
    med <- median(x)
    y <- 1*(x <= a)*(beta[1]*(x - med) - beta[1]*(a - med)) +
    1*(x >= 2158 - a)*(beta[2]*(x - med) - beta[2]*(2158 - a - med))
    return(data.frame("x" = x, "y" = y))
}

fake_groove <- sim_groove()
cp_gibbs2 <- detect_cp(data = fake_groove,
                    verbose = FALSE,
                    tol_edge = 50,
                    tol_cp = 1000,
                    iter = 300,
                    warmup = 100,
                    est_impute_par = FALSE)
```

---

example_data *Example of an average of 2D crosscuts from the Hamby 44 data set.*

Description

This data set is essentially a 2D crosscut from a land in the Hamby 44 set of bullet land scans.

Usage

```
example_data
```

Format

A data frame with 3346 rows and two variables:

x location variable

value the height of the land at the given location described by x

Source

Hamby 44

get_grooves_bcp *Conforming get_grooves_"name" function.*

Description

This is a wrapper function that comforms to the other get_grooves functions.

Usage

```
get_grooves_bcp(x, value, adjust = 10, ...)
```

Arguments

x	numeric vector of locations in microns
value	numeric vector of surface measurements in microns
adjust	positive number to adjust the grooves - XXX should be expressed in microns rather than an index
...	Additional arguments to be passed to detect_cp_v2.

Value

A named list containing the output from variable_cp_gibbs function, the range of data that was actually used for the changepoint algorithm (since it doesn't impute values past the outermost non-missing values), and the estimated groove locations.

Examples

```
data("example_data")
head(example_data)
example_data <- example_data[seq(from = 1, to = nrow(example_data), by = 30),]
cp_gibbs3 <- get_grooves_bcp(x = example_data$x,
    value = example_data$value,
    adjust = 10,
    iter = 300,
    warmup = 100)
```

imputeGP *Impute missing data.*

Description

This function imputes missing data based on a Gaussian process regression

Usage

```
imputeGP(y, x, sigma, l)
```

Arguments

y	Numeric y vector of response values.
x	Numeric x vector of locations used for the covariance function.
sigma	Marginal standard deviation in the Gaussian process.
l	Length scale parameter in the Gaussian process.

Value

A data frame with columns "x" and "y" which contain the combined observed and imputed data.

Examples

```
# Fake data
sim_groove <- function(beta = c(-0.28,0.28), a = 125)
{
    x <- seq(from = 0, to = 2158, by = 20)
    med <- median(x)
    y <- 1*(x <= a)*(beta[1]*(x - med) - beta[1]*(a - med)) +
    1*(x >= 2158 - a)*(beta[2]*(x - med) - beta[2]*(2158 - a - med))
    return(data.frame("x" = x, "y" = y))
}

fake_groove <- sim_groove()
fake_groove <- fake_groove[sample.int(n = nrow(fake_groove),
    size = round(0.8 * nrow(fake_groove)),
    replace = FALSE),]
fake_groove <- fake_groove[order(fake_groove$x),]
plot(fake_groove$x, fake_groove$y)

# add NA values where the data are missing
x_na <- seq(from = min(fake_groove$x), to = max(fake_groove$x),
 by = min(fake_groove$x[2:nrow(fake_groove)] - fake_groove$x[1:(nrow(fake_groove) - 1)]))
x_na <- x_na[!round(x_na, digits = 2) %in% round(fake_groove$x, digits = 2)]
y_na <- rep(NA, times = length(x_na))
d_na <- data.frame("x" = x_na, "y" = y_na)
fake_groove <- rbind(fake_groove, d_na)
fake_groove <- fake_groove[order(fake_groove$x),]

## impute the data
full_data <- imputeGP(y = fake_groove$y, x = fake_groove$x, sigma = 0.9, l = 15)
head(full_data)
plot(full_data$x, full_data$y)
```

---

mlgp *Impute missing data.*

Description

This function performs maximum likelihood estimation to estimate the variance parameters in a Gaussian process with a squared exponential covariance function. These parameters could then be used in the Gaussian process used for imputation.

Usage

```
mlgp(y, x, tol = 1e-06)
```

Arguments

y	Numeric y vector of response values.
x	Numeric x vector of locations used for the covariance function.
tol	Tolerance level for the maximum likelihood procedure to fit the Gaussian process.

Value

Standard optim output. The first optimized parameter value is the standard deviation the second is the length scale.

Examples

```
# Fake data
sim_groove <- function(beta = c(-0.28,0.28), a = 125)
{
    x <- seq(from = 0, to = 2158, by = 20)
    med <- median(x)
    y <- 1*(x <= a)*(beta[1]*(x - med) - beta[1]*(a - med)) +
    1*(x >= 2158 - a)*(beta[2]*(x - med) - beta[2]*(2158 - a - med))
    return(data.frame("x" = x, "y" = y))
}

fake_groove <- sim_groove()
fake_groove <- fake_groove[sample.int(n = nrow(fake_groove),
    size = round(0.8 * nrow(fake_groove)),
    replace = FALSE),]
plot(fake_groove$x, fake_groove$y)


# estimate the MLE's
mles <- mlgp(y = fake_groove$y, x = fake_groove$x)
```

`robust_loess_fit` *Fit a robust loess regression*

Description

Internal function called by get_grooves_lassobasic and get_grooves_lassofull

Usage

```
robust_loess_fit(cc, iter)
```

Arguments

`cc`	data frame with columns x and value_std, representing the crosscut
`iter`	number of iterations

Examples

```
data("example_data")
head(example_data)
example_data <- example_data[seq(from = 1, to = nrow(example_data), by = 30),]
plot(example_data$x, example_data$y)

# set the minimum y-value to zero
check_min <- min(example_data$value[!is.na(example_data$value)])
example_data <- dplyr::mutate(example_data, value_std = value - check_min)

# remove global structure
rlo_fit <- robust_loess_fit(cc = example_data, iter = 20)
example_data$rlo_pred <- predict(rlo_fit, newdata = example_data)
example_data$rlo_resid <- example_data$value_std - example_data$rlo_pred

# define new data frame without the global structure
data <- data.frame("x" = example_data$x, "y" = example_data$rlo_resid)
plot(data$x, data$y)
```

`runmcmc_cp0` *Estimate a posterior distribution of data conditional on zero change-points.*

Description

This function runs a random walk Metropolis algorithm to estimate the posterior distribution of a zero mean multivariate normal distribution with an covariance matrix generated by the exponential covariance function. This functions assumes equally spaced locations ("x" values in the "data" argument).

Usage

```
runmcmc_cp0(data, iter, start.vals, prop_var, warmup = 500, verbose = FALSE)
```

Arguments

data	Data frame with columns "x" and "y." "x" is a column of the locations of the observed residual values, y.
iter	Number of interations after warmup.
start.vals	List with elements "sigma" and "l" for the standard deviation and length scale which parameterize the covariance matrix.
prop_var	The proposal variance-covariance matrix for the random walk metropolis algorithm.
warmup	The number of initial iterations which serves two purposes: the first is to allow the algorithm to wander to the area of most mass, and the second is to tune the proposal variance.
verbose	Logical value indicating whether to print the iteration number and the parameter proposals.

Value

A named list. "parameters" is a list of named parameter values each of which is a vector of length "iter". "accept" gives the proportion of accepted proposals after warmup. "lp" is a vector of values of the log data pdf at each sampled parameter value.

Examples

```
# Fake data
sim_groove <- function(beta = c(-0.28,0.28), a = 125)
{
    x <- seq(from = 0, to = 2158, by = 20)
    med <- median(x)
    y <- 1*(x <= a)*(beta[1]*(x - med) - beta[1]*(a - med)) +
    1*(x >= 2158 - a)*(beta[2]*(x - med) - beta[2]*(2158 - a - med))
    return(data.frame("x" = x, "y" = y))
}

fake_groove <- sim_groove()
# define starting values
start.vals <- list("sigma" = c(1), "l" = c(10))

# proposal variance for the MH step
prop_var <- diag(c(1/2,1/2))

set.seed(1111)
m0cp <- runmcmc_cp0(data = fake_groove, iter = 500,
    start.vals = start.vals,
    prop_var = prop_var, warmup = 100, verbose = FALSE)
```

`runmcmc_cp1` *Estimate a posterior distribution of data conditional that there is one groove.*

Description

This function is basically a wrapper for running the left and right (one) changepoint Gibbs algorithms. The only computation that this function does is to estimate the posterior means of the left and right changepoint distributions.

Usage

```
runmcmc_cp1(
  data,
  iter,
  start.vals.left,
  start.vals.right,
  prop_var_left,
  prop_var_right,
  cp_prop_var,
  tol_edge = 10,
  warmup = 500,
  verbose = FALSE
)
```

Arguments

`data`	Data frame with columns "x" and "y." "x" is a column of the locations of the observed residual values, y.
`iter`	Number of interations after warmup.
`start.vals.left`	Starting values for the changepoint algorithm assuming the groove is on the left. List with elements "sigma", "l", "cp", "beta", and "intercept." "sigma" and "l" are 2 element vectors where the first element is for the data to the left of the changepoint. "cp" is the changepoint starting value. "beta" and "intercept" are the slope and intercept starting values for the mean of the data model to the left of the changepoint. which parameterize the covariance matrix.
`start.vals.right`	Starting values for the changepoint algorithm assuming the groove is on the right.
`prop_var_left`	The proposal variance for the random walk Metropolis algorithm assuming that the groove is on the left. A two element list of the proposal variance-covariance matrices for the random walk metropolis algorithm(s). The first element is for the data to the left of the changepoint.
`prop_var_right`	The proposal variance for the random walk Metropolis algorithm assuming that the groove is on the right.

cp_prop_var	The proposal variance for the changepoint.
tol_edge	This parameter controls how close changepoint proposals can be to the edge of the data before getting automatically rejected. For example, a value of 10 means that the changepoint will be automatically rejected if the proposal is within a distance of 10 x-values from either edge.
warmup	The number of initial iterations which serves two purposes: the first is to allow the algorithm to wander to the area of most mass, and the second is to tune the proposal variance.
verbose	Logical value indicating whether to print the iteration number and the parameter proposals.

Value

A named list with all of the output that the left and right changepoint functions produce individually plus the posterior means of the left and right changepoints.

Examples

```
# Fake data
sim_groove <- function(beta = c(-0.28,0.28), a = 125)
{
    x <- seq(from = 0, to = 2158, by = 20)
    med <- median(x)
    y <- 1*(x <= a)*(beta[1]*(x - med) - beta[1]*(a - med)) +
    1*(x >= 2158 - a)*(beta[2]*(x - med) - beta[2]*(2158 - a - med))
    return(data.frame("x" = x, "y" = y))
}

fake_groove <- sim_groove()

# define starting values for the changepoints
cp_start_left <- min(fake_groove$x) + 60
cp_start_right <- max(fake_groove$x) - 60

# define list of starting values for both the left and right changepoint models
start.vals <- list("left" = list("sigma" = c(1,1),
                                 "l" = c(10,10),
                                 "cp" = c(cp_start_left),
                                 "beta" = c(-1),
                                 "intercept" = c(0)),
                                 "right" = list("sigma" = c(1,1),
                                  "l" = c(10,10),
                                   "cp" = c(cp_start_right),
                                   "beta" = c(1),
                                   "intercept" = c(0)))

# list of starting values for each of the two MH steps
#  (not sampling the changepoint) for both the left and right changepoint models

prop_var <- list("left" = list(diag(c(1/2,1/2,1/2,1/2)),
                               diag(c(1/2,1/2))),
```

```
                          "right" = list(diag(c(1/2,1/2)),
                          diag(c(1/2,1/2,1/2, 1/2))))

# define the proposal variance for the RWMH step sampling the changepoint
cp_prop_var <- 10^2

# run Gibbs MCMC for both the right only and left only GEA models
set.seed(1111)
m1cp <- runmcmc_cp1(data = fake_groove, iter = 500,
                 start.vals.left = start.vals$left,
                 start.vals.right = start.vals$right,
                 prop_var_left = prop_var$left,
                 prop_var_right = prop_var$right,
                 cp_prop_var = cp_prop_var,
                 tol_edge = 50,
                 warmup = 100, verbose = FALSE)
```

`runmcmc_cp1_left` *Estimate a posterior distribution of data conditional on a left groove and no right groove.*

Description

This function runs a random walk metropolis within Gibbs algorithm to estimate the posterior distribution of the value of the changepoint as well as the parameters fit in each multivariate normal distribution on either side of the changepoint. The covariance matrices are both based on the exponential covariance function. This functions assumes equally spaced locations ("x" values in the "data" argument). The distribution to the left of the changepoint has a mean that is a linear function of the distance from the center of the data.

Usage

```
runmcmc_cp1_left(
  data,
  iter,
  start.vals,
  prop_var,
  cp_prop_var,
  tol_edge = 50,
  warmup = 500,
  verbose = FALSE
)
```

Arguments

`data`	Data frame with columns "x" and "y." "x" is a column of the locations of the observed residual values, y.
`iter`	Number of interations after warmup.

start.vals	List with elements "sigma", "l", "cp", "beta", and "intercept." "sigma" and "l" are 2 element vectors where the first element is for the data to the left of the changepoint. "cp" is the changepoint starting value. "beta" and "intercept" are the slope and intercept starting values for the mean of the data model to the left of the changepoint. which parameterize the covariance matrix.
prop_var	A two element list of the proposal variance-covariance matrices for the random walk metropolis algorithm(s). The first element is for the data to the left of the changepoint.
cp_prop_var	The proposal variance for the changepoint.
tol_edge	This parameter controls how close changepoint proposals can be to the edge of the data before getting automatically rejected. For example, a value of 10 means that the changepoint will be automatically rejected if the proposal is within a distance of 10 x-values from either edge.
warmup	The number of initial iterations which serves two purposes: the first is to allow the algorithm to wander to the area of most mass, and the second is to tune the proposal variance.
verbose	Logical value indicating whether to print the iteration number and the parameter proposals.

Value

A named list. "parameters" is a list of named parameter values each of which is a vector of length "iter". "accept" gives the proportion of accepted proposals after warmup. "lp" is a vector of values of the log data pdf at each sampled parameter value. "gp_prop_var" and "cp_prop_var" are the tuned proposal variances for the metropolis steps.

Examples

```
# Fake data
sim_groove <- function(beta = c(-0.28,0.28), a = 125)
{
    x <- seq(from = 0, to = 2158, by = 20)
    med <- median(x)
    y <- 1*(x <= a)*(beta[1]*(x - med) - beta[1]*(a - med)) +
    1*(x >= 2158 - a)*(beta[2]*(x - med) - beta[2]*(2158 - a - med))
    return(data.frame("x" = x, "y" = y))
}

fake_groove <- sim_groove()

# define starting values for the changepoints
cp_start_left <- min(fake_groove$x) + 60
cp_start_right <- max(fake_groove$x) - 60

# define list of starting values for both the left and right changepoint models
start.vals <- list("left" = list("sigma" = c(1,1),
                                 "l" = c(10,10),
                                 "cp" = c(cp_start_left),
                                 "beta" = c(-1),
```

```
                              "intercept" = c(0)),
                              "right" = list("sigma" = c(1,1),
                               "l" = c(10,10),
                                "cp" = c(cp_start_right),
                                "beta" = c(1),
                                "intercept" = c(0)))

# list of starting values for each of the two MH steps
# (not sampling the changepoint) for both the left and right changepoint models

prop_var <- list("left" = list(diag(c(1/2,1/2,1/2,1/2)),
                               diag(c(1/2,1/2))),
                               "right" = list(diag(c(1/2,1/2)),
                               diag(c(1/2,1/2,1/2, 1/2))))

# define the proposal variance for the RWMH step sampling the changepoint
cp_prop_var <- 10^2

# run Gibbs MCMC for the left GEA model
set.seed(1111)
m1cp_left <- runmcmc_cp1_left(data = fake_groove, iter = 500, warmup = 100,
                              start.vals = start.vals$left,
                              prop_var = prop_var$left,
                              cp_prop_var = cp_prop_var,
                              verbose = FALSE, tol_edge = 50)
```

`runmcmc_cp1_right` *Estimate a posterior distribution of data conditional on a left groove and no right groove.*

Description

This function runs a random walk metropolis within Gibbs algorithm to estimate the posterior distribution of the value of the changepoint as well as the parameters fit in each multivariate normal distribution on either side of the changepoint. The covariance matrices are both based on the exponential covariance function. This functions assumes equally spaced locations ("x" values in the "data" argument). The distribution to the right of the changepoint has a mean that is a linear function of the distance from the center of the data. Note that this function is identical to the 1cp_left function, and more thorough documentation is in that file.

Usage

```
runmcmc_cp1_right(
  data,
  iter,
  start.vals,
  prop_var,
  cp_prop_var,
  tol_edge = 50,
```

```
  warmup = 500,
  verbose = FALSE
)
```

Arguments

data	Data frame with columns "x" and "y." "x" is a column of the locations of the observed residual values, y.
iter	Number of interations after warmup.
start.vals	List with elements "sigma", "l", "cp", "beta", and "intercept." "sigma" and "l" are 2 element vectors where the first element is for the data to the left of the changepoint. "cp" is the changepoint starting value. "beta" and "intercept" are the slope and intercept starting values for the mean of the data model to the left of the changepoint. which parameterize the covariance matrix.
prop_var	A two element list of the proposal variance-covariance matrices for the random walk metropolis algorithm(s). The first element is for the data to the left of the changepoint.
cp_prop_var	The proposal variance for the changepoint.
tol_edge	This parameter controls how close changepoint proposals can be to the edge of the data before getting automatically rejected. For example, a value of 10 means that the changepoint will be automatically rejected if the proposal is within a distance of 10 x-values from either edge.
warmup	The number of initial iterations which serves two purposes: the first is to allow the algorithm to wander to the area of most mass, and the second is to tune the proposal variance.
verbose	Logical value indicating whether to print the iteration number and the parameter proposals.

Value

A named list. "parameters" is a list of named parameter values each of which is a vector of length "iter". "accept" gives the proportion of accepted proposals after warmup. "lp" is a vector of values of the log data pdf at each sampled parameter value. "gp_prop_var" and "cp_prop_var" are the tuned proposal variances for the metropolis steps.

Examples

```
# Fake data
sim_groove <- function(beta = c(-0.28,0.28), a = 125)
{
    x <- seq(from = 0, to = 2158, by = 20)
    med <- median(x)
    y <- 1*(x <= a)*(beta[1]*(x - med) - beta[1]*(a - med)) +
    1*(x >= 2158 - a)*(beta[2]*(x - med) - beta[2]*(2158 - a - med))
    return(data.frame("x" = x, "y" = y))
}

fake_groove <- sim_groove()
```

```
# define starting values for the changepoints
cp_start_left <- min(fake_groove$x) + 60
cp_start_right <- max(fake_groove$x) - 60

# define list of starting values for both the left and right changepoint models
start.vals <- list("left" = list("sigma" = c(1,1),
                                 "l" = c(10,10),
                                 "cp" = c(cp_start_left),
                                 "beta" = c(-1),
                                 "intercept" = c(0)),
                                 "right" = list("sigma" = c(1,1),
                                  "l" = c(10,10),
                                   "cp" = c(cp_start_right),
                                   "beta" = c(1),
                                   "intercept" = c(0)))

# list of starting values for each of the two MH steps
# (not sampling the changepoint) for both the left and right changepoint models

prop_var <- list("left" = list(diag(c(1/2,1/2,1/2,1/2)),
                               diag(c(1/2,1/2))),
                               "right" = list(diag(c(1/2,1/2)),
                               diag(c(1/2,1/2,1/2, 1/2))))

# define the proposal variance for the RWMH step sampling the changepoint
cp_prop_var <- 10^2

# run Gibbs MCMC for the right GEA model
set.seed(1111)
m1cp_right<- runmcmc_cp1_right(data = fake_groove, iter = 500, warmup = 100,
                               start.vals = start.vals$right,
                               prop_var = prop_var$right,
                               cp_prop_var = cp_prop_var,
                               verbose = FALSE, tol_edge = 50)
```

---

runmcmc_cp2 *Estimate a posterior distribution of data conditional that there are two grooves.*

Description

This function runs a random walk metropolis within Gibbs algorithm to estimate the posterior distribution of the value of the changepoints as well as the parameters fit in each multivariate normal distribution on either side of each changepoint. The covariance matrices are based on the exponential covariance function. This functions assumes equally spaced locations ("x" values in the "data" argument). The distribution to the right of the right most changepoint and to the left of the left most changepoint have means that are a linear function of the distance from the center of the data. The slope is constrained to be negative in the left case and positive in the right case. The models fit to the groove engraved areas are exactly the same as in the one changepoint case. Thus, this algorithm only differs in that there are three segments of data to deal with as opposed to two.

Usage

```
runmcmc_cp2(
  data,
  iter,
  start.vals,
  prop_var,
  cp_prop_var,
  tol_edge = 50,
  tol_cp = 1000,
  warmup = 500,
  verbose = FALSE
)
```

Arguments

data	Data frame with columns "x" and "y." "x" is a column of the locations of the observed residual values, y.
iter	Number of interations after warmup.
start.vals	Starting values for the changepoint algorithm. List with elements "sigma", "l", "cp", "beta", and "intercept." "sigma" and "l" are 3 element vectors where the first element is for the data on the left groove. The second element is for the land engraved area, and the third element is for the right groove. "cp" is the vector of changepoint starting values. "beta" and "intercept" are two element vectors of the slope and intercept for the left and right groove engraved area respectively.
prop_var	A three element list of the proposal variance-covariance matrices for the random walk Metropolis algorithm(s). The first element is for the left groove engraved area. The second element is for the land engraved area, and the third element is for the right engraved area.
cp_prop_var	The proposal variance-covariance matrix for the changepoints.
tol_edge	This parameter controls how close changepoint proposals can be to the edge of the data before getting automatically rejected. For example, a value of 10 means that the changepoint will be automatically rejected if either of the proposal changepoints is within a distance of 10 x-values from either edge.
tol_cp	This parameter controls how close changepoint proposals can be to each other before getting automatically rejected. For example, a value of 10 means that the changepoint will be automatically rejected if either of the proposal changepoints is within a distance of 10 x-values from either each other.
warmup	The number of initial iterations which serves two purposes: the first is to allow the algorithm to wander to the area of most mass, and the second is to tune the proposal variance.
verbose	Logical value indicating whether to print the iteration number and the parameter proposals.

Value

A named list containing the sampled parameters, acceptance rates for the Metropolis steps, log likelihood values, and proposal variance for the changepoints.

Examples

```
# Fake data
sim_groove <- function(beta = c(-0.28,0.28), a = 125)
{
    x <- seq(from = 0, to = 2158, by = 20)
    med <- median(x)
    y <- 1*(x <= a)*(beta[1]*(x - med) - beta[1]*(a - med)) +
    1*(x >= 2158 - a)*(beta[2]*(x - med) - beta[2]*(2158 - a - med))
    return(data.frame("x" = x, "y" = y))
}

fake_groove <- sim_groove()

# define starting values for the changepoints
cp_start_left <- min(fake_groove$x) + 60
cp_start_right <- max(fake_groove$x) - 60

# define starting values
start.vals <- list("sigma" = c(1,1,1),
                   "l" = c(10,10,10),
                   "cp" = c(cp_start_left, cp_start_right),
                   "beta" = c(-2,2),
                   "intercept" = c(0,0))

# define proposal variances (not for changepoints)
prop_var <- list(diag(c(1/2,1/2,1/2,1/2)),
                 diag(c(1/2,1/2)),
                 diag(c(1/2,1/2,1/2,1/2)))

# define proposal variance for changepoints
cp_prop_var <- diag(c(10^2, 10^2))


# run Gibbs MCMC for both the right only and left only GEA models
set.seed(1111)
m2cp <- runmcmc_cp2(data = fake_groove,
                    iter = 500,
                    start.vals = start.vals,
                    prop_var = prop_var,
                    cp_prop_var = cp_prop_var,
                    tol_edge = 50, tol_cp = 1000,
                    warmup = 100,
                    verbose = FALSE)
```

`runmcmc_cpall` *Estimate posterior distributions for the 0, 1, or 2 changepoint case.*

Description

This function runs the changepoint functions designed for the cases when there are 0, 1, or 2 changepoints. It then returns a subset of the results that are returned for each function individually. This

subset of results is enough to decide the likely number of shoulders, the locations of the shoulders (if they exist), as well as the posterior samples for the changepoints for minimal diagnostic use.

Usage

```
runmcmc_cpall(
  data,
  iter = 8000,
  start.vals = NA,
  prop_var = NA,
  cp_prop_var = NA,
  tol_edge = 50,
  tol_cp = 1000,
  warmup = 500,
  verbose = FALSE,
  prior_numcp = rep(1/4, times = 4)
)
```

Arguments

data	Data frame with columns "x" and "y." "x" is a column of the locations of the observed residual values, y.
iter	Number of iterations after warmup.
start.vals	Starting values for the changepoint algorithm. Either NA valued or a named list of lists. If list, the names of the lists should be "cp2","cp1", and "cp0". Each list posessing one of those aforementioned names is a list of starting values identical to what would be given if the changepoint algorithm were to be run with the corresponding number of specified changepoints. List with elements "sigma", "l", "cp", "beta", and "intercept." "sigma" and "l" are 3 element vectors where the first element is for the data on the left groove. The second element is for the land engraved area, and the third element is for the right groove. "cp" is the vector of changepoint starting values. "beta" and "intercept" are two element vectors of the slope and intercept for the left and right groove engraved area respectively. If NA, default starting values will be used. Note that the changepoint starting values should always be near the edges of the data.
prop_var	Either NA valued or a list of named lists. If list, the names of the lists should be "cp2","cp1", and "cp0". Each list posessing one of those aforementioned names is a list of proposal covariance matrices identical to what would be given if the changepoint algorithm were to be run with the corresponding number of specified changepoints.
cp_prop_var	The proposal variance-covariance matrix for the changepoints. Can either be NA or a named list. If list, the names of the list items should be "cp2", "cp1" where each is the appropriate proposal variance/covariance matrix for the number of changepoints.
tol_edge	This parameter controls how close changepoint proposals can be to the edge of the data before getting automatically rejected. For example, a value of 10 means that the changepoint will be automatically rejected if either of the proposal changepoints is within a distance of 10 x-values from either edge.

tol_cp	This parameter controls how close changepoint proposals can be to each other before getting automatically rejected. For example, a value of 10 means that the changepoint will be automatically rejected if either of the proposal changepoints is within a distance of 10 x-values from either each other.
warmup	The number of warmup iterations. This should be set to a very small number of iterations, as using too many iterations as warmup risks moving past the changepoints and getting stuck in a local mode. Default is set to 500.
verbose	Logical value indicating whether to print the iteration number and the parameter proposals.
prior_numcp	This is a vector with four elements giving the prior probabilities for the zero changepoint model, the one changepoint on the left model, the one changepoint on the right model, and the two changepoint model, in that order. Note that, practically, because the likelihood values are so large, only very strong priors will influence the results.

Value

A named list containing the sampled changepoint locations for both the one and two changepoint scenarios, the posterior changepoint means, the average log pdf values from the data model under each model, the maximum log probability values under each model log likelihood values, and estimates of the maximum a posteriori changepoint value under each model.

Examples

```
# Fake data
sim_groove <- function(beta = c(-0.28,0.28), a = 125)
{
    x <- seq(from = 0, to = 2158, by = 20)
    med <- median(x)
    y <- 1*(x <= a)*(beta[1]*(x - med) - beta[1]*(a - med)) +
    1*(x >= 2158 - a)*(beta[2]*(x - med) - beta[2]*(2158 - a - med))
    return(data.frame("x" = x, "y" = y))
}

fake_groove <- sim_groove()

# define starting values for the changepoints
cp_start_left <- min(fake_groove$x) + 60
cp_start_right <- max(fake_groove$x) - 60

# define list of starting values for both the left and right changepoint models
cp0.start.vals <- list("sigma" = c(1), "l" = c(10))
cp1.start.vals <- list("left" = list("sigma" = c(1,1),
                                "l" = c(10,10),
                                "cp" = c(cp_start_left),
                                "beta" = c(-1),
                                "intercept" = c(0)),
                                "right" = list("sigma" = c(1,1),
                                 "l" = c(10,10),
                                  "cp" = c(cp_start_right),
```

```
                                "beta" = c(1),
                                "intercept" = c(0)))
cp2.start.vals <- list("sigma" = c(1,1,1),
                "l" = c(10,10,10),
                "cp" = c(cp_start_left, cp_start_right),
                "beta" = c(-2,2),
                "intercept" = c(0,0))
start.vals <- list("cp2" = cp2.start.vals, "cp1" = cp1.start.vals, "cp0" = cp0.start.vals)

# list of starting values for each of the two MH steps
# (not sampling the changepoint) for both the left and right changepoint models
prop_var_0cp <- diag(c(1/2,1/2))
prop_var_lrcp <- list("left" = list(diag(c(1/2,1/2,1/2,1/2)),
                            diag(c(1/2,1/2))),
                            "right" = list(diag(c(1/2,1/2)),
                            diag(c(1/2,1/2,1/2, 1/2))))

prop_var_2cp <- list(diag(c(1/2,1/2,1/2,1/2)),
              diag(c(1/2,1/2)),
              diag(c(1/2,1/2,1/2,1/2)))

prop_var <- list("cp2" = prop_var_2cp, "cp1" = prop_var_lrcp, "cp0" = prop_var_0cp)

# define the proposal variance for the RWMH step sampling the changepoint
cp_prop_var <- list("cp2" = diag(c(10^2, 10^2)),
                 "cp1" = 10^2)

# prior on the number of changepoints
prior_numcp <- rep(1/4, times = 4)

set.seed(1111)
cp_gibbs <- runmcmc_cpall(data = fake_groove,
                       start.vals = start.vals,
                       prop_var = prop_var,
                       cp_prop_var = cp_prop_var,
                       verbose = FALSE,
                       tol_edge = 50,
                       tol_cp = 1000,
                       iter = 300,
                       warmup = 100,
                       prior_numcp = prior_numcp)
```

Index

∗ **datasets**
example_data, 4

detect_cp, 2

example_data, 4

get_grooves_bcp, 5

imputeGP, 5

mlgp, 6

robust_loess_fit, 8
runmcmc_cp0, 8
runmcmc_cp1, 10
runmcmc_cp1_left, 12
runmcmc_cp1_right, 14
runmcmc_cp2, 16
runmcmc_cpall, 18