Package 'bcp'

January 27, 2015

Type Package

Title A Package for Performing a Bayesian Analysis of Change Point Problems

Version 3.0.1

Date 2012-04-17

Author Chandra Erdman and John W. Emerson

Maintainer John W. Emerson <john.emerson@yale.edu>

Depends methods, stats, graphics, foreach, iterators, grid, Rcpp (>= 0.9.2)

Suggests DNAcopy, coda, strucchange

Description This package provides an implementation of an approximation to the Barry and Hartigan (1993) product partition model for the normal errors change point problem using Markov Chain Monte Carlo. It also extends the methodology to independent multivariate series with an assumed common change point structure.

License GPL (>= 2)

LinkingTo Rcpp

Repository CRAN

Date/Publication 2012-04-19 07:28:48

NeedsCompilation yes

R topics documented:

bcp *A Package for Performing a Bayesian Analysis of Change Point Problems*

Description

`bcp()` implements an approximation to the Barry and Hartigan (1993) product partition model for the normal errors change point problem using Markov Chain Monte Carlo; it also offers an extension of the model to the multivariate case. This algorithm is used when there exists an unknown partition of a sequence, or sequences, into contiguous blocks such that the mean is constant within each block. In the multivariate case, means are constant within each block of each sequence, but may differ across sequences within a given block. Conditional on the partition, the model assumes that observations are independent, identically distributed normal, with constant means within blocks and constant variance throughout each sequence. The original methodology is extended to allow multivariate analysis; when multivariate series are available, a common change point structure is assumed, but series may have different means within a block.

Usage

```
bcp(x, w0 = 0.2, p0 = 0.2, burnin = 50, mcmc = 500,
    return.mcmc = FALSE)
```

Arguments

`x`	a vector or matrix of numerical data (with no missing values). For the multivariate change point problems, each column corresponds to a series.
`w0`	an optional numeric value for the prior, $U(0, w0)$, on the signal-to-noise ratio. If no value is specified, the default value of 0.2 is used, as recommended by Barry and Hartigan (1993).
`p0`	an optional numeric value for the prior, $U(0, p0)$, on the probability of a change point at each location in the sequence. If no value is specified, the default value of 0.2 is used, as recommended by Barry and Hartigan (1993).
`burnin`	the number of burnin iterations.
`mcmc`	the number of iterations after burnin.
`return.mcmc`	if `return.mcmc=TRUE` the posterior means and the partitions after each iteration are returned.

Details

This algorithm is used when there exists an unknown partition of a sequence, or sequences, into contiguous blocks such that the mean is constant within each block (and each block of each sequence in the multivariate case). The primary result is an estimate of the posterior mean (or its distribution if `return.mcmc` is `TRUE`) at every location. Unlike a frequentist or algorithmic approach to the problem, these estimates will not be constant within regions, and no single partition is identified as best. Estimates of the probability of a change point at any given location are provided, however.

The user may set `.Random.seed` to control the MCMC iterations.

The functions `summary.bcp`, `print.bcp`, and `plot.bcp` are used to obtain summaries of the results; `legacyplot` is included from package versions prior to 3.0.0 and will only work for univariate change point analyses.

If there is a registered parallel backend (probably via **doSNOW**, **doMC**, or **doMPI**) then parallel Markov chains will be run on the available resources. There is communication overhead as well as the overhead associated with burning in each chain.

A special note is needed about the values returned and possible parallel computation. First, `blocks` contains a count of the number of blocks in the partition at any given iteration of the MCMC procedure. Similarly, `mcmc.means` and `mcmc.rhos` contain information on the means and current change point locations for any iteration. In the univariate case, when `mcmc.return=TRUE`, the burnins (or multiple sets of burnins if run in parallel) are collected at the beginning of the matrix of results, followed by blocks of the mcmc results. So, for example, positions `1:burnin` contain the burnins from the first worker. And if there are `k` workers, then positions `k*burnin+1` through `k*burnin+mcmc/k`, roughly, will contain the results of the first chain (ideally `mcmc` should be divisible by `k`). As a result, convergence diagnostics (for example, the `heidel.diag()` function of the `coda` package) should be applied to parallel `bcp` objects with care. In the multivariate case, the burnins and subsequent mcmc iteration results are separated into lists for convenience.

Value

`bcp()` returns a list containing the following components:

`data`	a copy of the data.
`return.mcmc`	`TRUE` or `FALSE` as specified by the user; see the arguments, above.
`mcmc.means`	if `return.mcmc=TRUE`, `mcmc.means` contains the means for each iteration conditional on the state of the partition.
`mcmc.rhos`	if `return.mcmc=TRUE`, `mcmc.rhos` contains the partitions after each iteration. A value of 1 indicates the end of a block.
`blocks`	a vector of the number of blocks after each iteration.
`posterior.mean`	a vector or matrix of the estimated posterior means.
`posterior.var`	a vector or matrix of the estimated posterior variances.
`posterior.prob`	a vector of the estimated posterior probabilities of changes at each location.
`burnin`	the number of burnin iterations.
`mcmc`	the number of iterations after burnin.
`w0`	see the arguments, above.
`p0`	see the arguments, above.

Note

Versions < 2.0 are quadratic in speed, and perform the default 550 iterations in approximately 0.75 seconds for a sequence of length 100. Versions >= 2.0 are linear in speed and partition a sequence of length 10,000 in approximately 45 seconds (compared with 45 minutes for versions < 2.0). These times were computed on a PC with Windows XP, a Pentium D Processor (2.99 GHz) and 3.50GB of RAM. Versions < 2.2.0 used NetWorkSpaces for optional parallel mcmc. Versions >= 2.2.0 replace this with the more flexible and friendly package **foreach**. Multivariate analysis is supported in versions >= 3.0.

Author(s)

Chandra Erdman and John W. Emerson

Maintainer: John W. Emerson <john.emerson@yale.edu>

References

J. Bai and P. Perron (2003), Computation and Analysis of Multiple Structural Change Models, *Journal of Applied Econometrics*, **18**, 1-22. http://qed.econ.queensu.ca/jae/2003-v18.1/bai-perron/.

Daniel Barry and J. A. Hartigan (1993), A Bayesian Analysis for Change Point Problems, *Journal of The American Statistical Association*, **88**, 309-19.

Chandra Erdman and John W. Emerson (2008), A Fast Bayesian Change Point Analysis for the Segmentation of Microarray Data, *Bioinformatics*, 24(19), 2143–2148. http://bioinformatics.oxfordjournals.org/cgi/content/abstract/btn404.

Chandra Erdman and John W. Emerson (2007), bcp: An R Package for Performing a Bayesian Analysis of Change Point Problems. *Journal of Statistical Software*, 23(3), 1–13. http://www.jstatsoft.org/v23/i03/.

A. B. Olshen, E. S. Venkatraman, R. Lucito, M. Wigler (2004), Circular binary segmentation for the analysis of array-based DNA copy number data, *Biostatistics*, **5**, 557-572. http://www.bioconductor.org/repository/release1.5/package/html/DNAcopy.html.

Snijders *et al.* (2001), Assembly of microarrays for genome-wide measurement of DNA copy number, *Nature Genetics*, **29**, 263-264.

Achim Zeileis, Friedrich Leisch, Kurt Hornik, Christian Kleiber (2002), strucchange: An R Package for Testing for Structural Change in Linear Regression Models, *Journal of Statistical Software*, **7**(2), 1–38. http://www.jstatsoft.org/v07/i02/.

See Also

plot.bcp, summary.bcp, and print.bcp for summaries of the results.

Examples

```
##### A random sample from a few normal distributions #####
testdata <- c(rnorm(50), rnorm(50, 5, 1), rnorm(50))
bcp.0 <- bcp(testdata)
plot.bcp(bcp.0)
legacyplot(bcp.0)

##### Coriell chromosome 11 #####
data(coriell)
chrom11 <- as.vector(na.omit(coriell$Coriell.05296[coriell$Chromosome==11]))
bcp.11 <- bcp(chrom11)
plot.bcp(bcp.11)

## Not run:
  ##### An example using foreach for parallel MCMC; note that
  ##### you must register a parallel backend using doSNOW, doMC,
```

```
    ##### or doMPI.  This example would use doSNOW.

    library(doSNOW)
    cl <- makeCluster(3, type="SOCK")
    registerDoSNOW(cl)

    # This probably takes around 3.5 seconds:
    system.time(bcp.par <- bcp(chrom11, mcmc=20000))
    stopCluster(cl)

    # This sequential run is slower:
    registerDoSEQ()  # The default behavior
    system.time(bcp.seq <- bcp(chrom11, mcmc=20000))

## End(Not run)

  # To see bcp and Circular Binary Segmentation results, using
  # base graphics (see plot.bcp.legacy for more examples):
  if(require("DNAcopy")) {
    n <- length(chrom11)
    cbs <- segment(CNA(chrom11, rep(1, n), 1:n), verbose = 0)
    cbs.ests <- rep(unlist(cbs$output[6]), unlist(cbs$output[5]))
    op <- par(mfrow=c(2,1),col.lab="black",col.main="black")
    op2 <- par(mar=c(0,4,4,2),xaxt="n", cex.axis=0.75)
    plot(1:n, bcp.11$data, col="grey", pch=20, xlab="Location",
         ylab="Posterior Mean",
         main="Posterior Means and Probabilities of a Change")
    lines(cbs.ests, col="red")
    lines(bcp.11$posterior.mean, lwd=2)
    par(op2)
    op3 <- par(mar=c(5,4,0,2), xaxt="s", cex.axis=0.75)
    plot(1:n, bcp.11$posterior.prob, type="l", ylim=c(0,1),
         xlab="Location", ylab="Posterior Probability", main="")
    for (i in 1:(dim(cbs$output)[1]-1)) {
      abline(v=cbs$output$loc.end[i], col="red")
    }
    par(op3)
    par(op)
  } else {
        cat("DNAcopy is not loaded")
  }

  ##### RealInt #####
  data("RealInt")
  bcp.ri <- bcp(as.vector(RealInt))
  plot.bcp(bcp.ri)

  # To see bcp and Bai and Perron results:
  if (require("strucchange")) {
    bp <- breakpoints(RealInt ~ 1, h = 2)$breakpoints
    rho <- rep(0, length(RealInt))
    rho[bp] <- 1
    b.num<-1 + c(0,cumsum(rho[1:(length(rho)-1)]))
```

```
  bp.mean <- unlist(lapply(split(RealInt,b.num),mean))
  bp.ri <- rep(0,length(RealInt))
  for (i in 1:length(bp.ri)) bp.ri[i] <- bp.mean[b.num[i]]
  xax <- seq(1961, 1987, length=103)
  op <- par(mfrow=c(2,1),col.lab="black",col.main="black")
  op2 <- par(mar=c(0,4,4,2),xaxt="n", cex.axis=0.75)
  plot(1:length(bcp.ri$data), bcp.ri$data, col="grey", pch=20,
       xlab="", ylab="Posterior Mean", main="U.S. Ex-Post Interest Rate")
  lines(bcp.ri$posterior.mean, lwd=2)
  lines(bp.ri, col="blue")
  par(op2)
  op3 <- par(mar=c(5,4,0,2), xaxt="s", cex.axis=0.75)
  plot(xax, bcp.ri$posterior.prob, yaxt="n", type="l", ylim=c(0,1),
       xlab="Year", ylab="Posterior Probability", main="")
  for (i in 1:length(bp.ri)) abline(v=xax[bp[i]], col="blue")
  axis(2, yaxp=c(0, 0.9, 3))
  par(op3)
  par(op)
} else {
  cat("strucchange is not loaded")
}

##### A multivariate example #####
testdata <- cbind( c(rnorm(50), rnorm(50, -5, 1), rnorm(50)),
                   c(rnorm(50), rnorm(50, 10.8, 1), rnorm(50, -3, 1)) )
bcp.0 <- bcp(testdata)
plot.bcp(bcp.0)
plot.bcp(bcp.0, separated=TRUE)
```

`coriell` *Array CGH data set of Coriell cell lines*

Description

These are two data array CGH studies of Corriel cell lines taken from the reference below.

Usage

```
data(coriell)
```

Format

A data frame containing five variables: first is clone name, second is clone chromosome, third is clone position, fourth and fifth are log2ratio for two cell lines.

Source

http://www.nature.com/ng/journal/v29/n3/suppinfo/ng754_S1.html

References

Olshen, A. B., Venkatraman, E. S., Lucito, R., Wigler, M. (2004), Circular binary segmentation for the analysis of array-based DNA copy number data, *Biostatistics*, **5**, 557-572. url: http://www.bioconductor.org/repository/release1.5/package/html/DNAcopy.html.

Snijders *et al.* (2001), Assembly of microarrays for genome-wide measurement of DNA copy number, *Nature Genetics*, **29**, 263-264.

Examples

```
##### Coriell chromosome 11 #####
data(coriell)
chrom11 <- na.omit(coriell$Coriell.05296[coriell$Chromosome==11])
n <- length(chrom11)
bcp.11 <- bcp(chrom11[1:n])
plot.bcp(bcp.11)

##### Coriell chromosome 11 #####
data(coriell)
chrom11 <- as.vector(na.omit(coriell$Coriell.05296[coriell$Chromosome==11]))
bcp.11 <- bcp(chrom11)
plot.bcp(bcp.11)

# To see bcp and Circular Binary Segmentation results:
if (require("DNAcopy")) {
  n <- length(chrom11)
  cbs <- segment(CNA(chrom11, rep(1, n), 1:n), verbose = 0)
  cbs.ests <- rep(unlist(cbs$output[6]), unlist(cbs$output[5]))
  op <- par(mfrow=c(2,1),col.lab="black",col.main="black")
  op2 <- par(mar=c(0,4,4,2),xaxt="n", cex.axis=0.75)
  plot(1:n, bcp.11$data, col="grey", pch=20, xlab="Location",
       ylab="Posterior Mean",
       main="Posterior Means and Probabilities of a Change")
  lines(cbs.ests, col="red")
  lines(bcp.11$posterior.mean, lwd=2)
  par(op2)
  op3 <- par(mar=c(5,4,0,2), xaxt="s", cex.axis=0.75)
  plot(1:n, bcp.11$posterior.prob, type="l", ylim=c(0,1),
       xlab="Location", ylab="Posterior Probability", main="")
  for (i in 1:(dim(cbs$output)[1]-1)) {
    abline(v=cbs$output$loc.end[i], col="red")
  }
  par(op3)
  par(op)
} else {
  cat("DNAcopy is not loaded")
}
```

`fitted.bcp` *Extract model fitted values*

Description

fitted method for class bcp.

Usage

```
 ## S3 method for class 'bcp'
fitted(object, ...)
```

Arguments

object	the result of a call to bcp().
...	additional arguments.

Value

Fitted values extracted from the bcp object.

Author(s)

Chandra Erdman and John W. Emerson

See Also

plot.bcp, summary.bcp, and print.bcp for summaries of the results.

Examples

```
##### A random sample from a few normal distributions #####
testdata <- c(rnorm(50), rnorm(50, 5, 1), rnorm(50))
bcp.0 <- bcp(testdata)
plot.bcp(bcp.0)
fitted.bcp(bcp.0)
```

---

interval.prob *Estimate the probability of a change point in a specified interval.*

Description

The function interval.prob() estimates the probability of at least one change point in the specified interval; it may only be used when return.mcmc=TRUE.

Usage

```
interval.prob(object, start, end)
```

Arguments

object	the result of a call to bcp().
start	the starting index of the interval.
end	the ending index of the interval.

Details

The function returns an estimate of the posterior probability of at least one change point in the specified interval.

Note

return.mcmc must be TRUE

Author(s)

Chandra Erdman and John W. Emerson

See Also

bcp and plot.bcp.

Examples

```
##### A random sample from a few normal distributions #####
testdata <- c(rnorm(50), rnorm(50, 5, 1), rnorm(50))
bcp.0 <- bcp(testdata, return.mcmc=TRUE)
plot.bcp(bcp.0)
interval.prob(bcp.0, 45, 55)
```

---

plot.bcp *Plotting Bayesian change point results*

Description

plot.bcp() produces summary plots of the results of bcp().

Usage

```
 ## S3 method for class 'bcp'
plot(x, separated = FALSE,
   outer.margins = list(left=unit(4, "lines"),
     bottom=unit(3, "lines"),
     right=unit(2, "lines"),
     top=unit(2, "lines")),
```

```
lower.area = unit(0.33, "npc"),
size.points = unit(0.25, "char"),
pch.points = 20,
colors = NULL,
main = NULL,
cex.axes = list(cex.xaxis = 0.75,
  cex.yaxis.lower = 0.75,
  cex.yaxis.upper.default = 0.75,
  cex.yaxis.upper.separated = 0.5),
 ...)
```

Arguments

x	the result of a call to bcp().
separated	if separated=TRUE and the data is multivariate, each series is plotted separately.
outer.margins	an optional list of units specifying the left, bottom, right and top margins. For more information on units, see the documentation for grid.
lower.area	an optional unit specifying the proportion of the plot occupied by the posterior probabilities of change points.
size.points	an optional unit specifying the size of the data points.
pch.points	an optional unit specifying the style of the data points.
colors	an optional vector specifying the colors in which to plot each data series.
main	an optional plot title. Use "" for no title.
cex.axes	an optional list specifying the sizes of the axes labels. cex.xaxis specifies the size of the x-axis label, cex.yaxis.lower specifies the size of the y-axis label of the posterior probability plot, cex.yaxis.upper.default specifies the size of the y-axis labels of the posterior means plot when the series are displayed in a single plot, and cex.yaxis.upper.separated specifies the size of the y-axis labels of the posterior means plots when each series is plotted separately.
...	additional arguments, ignored.

Details

plot.bcp() produces the following plots using grid graphics instead of base:

Posterior Means: location in the sequence versus the posterior means over the iterations.

Posterior Probability of a Change: location in the sequence versus the relative frequency of iterations which resulted in a change point.

Note

The original plot method for bcp from package versions prior to 3.0.0 is available in legacyplot; this plot method uses base graphics.

Author(s)

Chandra Erdman and John W. Emerson

See Also

legacyplot, bcp, summary.bcp, and print.bcp for complete results and summary statistics.

Examples

```
testdata <- cbind( c(rnorm(50), rnorm(50, -5, 1), rnorm(50)),
                   c(rnorm(50), rnorm(50, 10.8, 1), rnorm(50, -3, 1)) )
bcp.0 <- bcp(testdata)
plot.bcp(bcp.0)
plot.bcp(bcp.0, separated=TRUE)
```

---

plot.bcp.legacy *Plotting univariate Bayesian change point results*

Description

legacyplot() produces summary plots of the results of bcp() when used for univariate analysis; it was the default method prior to package version 3.0.0.

Usage

```
legacyplot(x, ...)
```

Arguments

x	the result of a call to bcp().
...	additional arguments.

Details

legacyplot() produces the following plots using base graphics:

Posterior Means: location in the sequence versus the posterior mean over the iterations.

Posterior Probability of a Change: location in the sequence versus the relative frequency of iterations which resulted in a change point.

Author(s)

Chandra Erdman and John W. Emerson

See Also

plot.bcp, bcp, summary.bcp, and print.bcp for complete results and summary statistics.

Examples

```
##### A random sample from a few normal distributions #####
testdata <- c(rnorm(50), rnorm(50, 5, 1), rnorm(50))
bcp.0 <- bcp(testdata)
legacyplot(bcp.0)
```

RealInt *US Ex-post Real Interest Rate data, 1961(1):1986(3)*

Description

US ex-post real interest rate: the three-month treasury bill deflated by the CPI inflation rate.

Usage

```
data("RealInt")
```

Format

A quarterly time series from 1961(1) to 1986(3).

Source

The data is available online in the data archive of the *Journal of Applied Econometrics*. url: http://qed.econ.queensu.ca/jae/2003-v18.1/bai-perron/.

References

J. Bai and P. Perron (2003), Computation and Analysis of Multiple Structural Change Models, *Journal of Applied Econometrics*, **18**, 1-22. http://qed.econ.queensu.ca/jae/2003-v18.1/bai-perron/.

Achim Zeileis, Friedrich Leisch, Kurt Hornik, Christian Kleiber (2002), strucchange: An R Package for Testing for Structural Change in Linear Regression Models, *Journal of Statistical Software*, **7**(2), 1–38. http://www.jstatsoft.org/v07/i02/.

Examples

```
 ##### RealInt #####
data("RealInt")
bcp.ri <- bcp(as.vector(RealInt))
plot.bcp(bcp.ri)

# to see bcp and Bai and Perron results run:
if (require("strucchange")) {
  bp <- breakpoints(RealInt ~ 1, h = 2)$breakpoints
  rho <- rep(0, length(RealInt))
```

```
  rho[bp] <- 1
  b.num<-1 + c(0,cumsum(rho[1:(length(rho)-1)]))
  bp.mean <- unlist(lapply(split(RealInt,b.num),mean))
  bp.ri <- rep(0,length(RealInt))
  for (i in 1:length(bp.ri)) bp.ri[i] <- bp.mean[b.num[i]]
    xax <- seq(1961, 1987, length=103)
    op<-par(mfrow=c(2,1),col.lab="black",col.main="black")
    op2 <- par(mar=c(0,4,4,2),xaxt="n", cex.axis=0.75)
    plot(1:length(bcp.ri$data), bcp.ri$data, col="grey", pch=20,
      xlab="", ylab="Posterior Mean", main="U.S. Ex-Post Interest Rate")
    lines(bcp.ri$posterior.mean, lwd=2)
    lines(bp.ri, col="blue")
    par(op2)
    op3 <- par(mar=c(5,4,0,2), xaxt="s", cex.axis=0.75)
    plot(xax, bcp.ri$posterior.prob, yaxt="n", type="l", ylim=c(0,1),
      xlab="Year", ylab="Posterior Probability", main="")
    for (i in 1:length(bp.ri)) abline(v=xax[bp[i]], col="blue")
    axis(2, yaxp=c(0, 0.9, 3))
    par(op3)
    par(op)
} else {
  cat("strucchange is not loaded")
}
```

`residuals.bcp` *Extract model residuals*

Description

residuals method for class `bcp`.

Usage

```
 ## S3 method for class 'bcp'
residuals(object, ...)
```

Arguments

`object`	the result of a call to `bcp()`.
`...`	additional arguments.

Value

Residuals extracted from the `bcp` object.

Author(s)

Chandra Erdman and John W. Emerson

See Also

bcp and plot.bcp.

Examples

```
##### A random sample from a few normal distributions #####
testdata <- c(rnorm(50), rnorm(50, 5, 1), rnorm(50))
bcp.0 <- bcp(testdata)
plot.bcp(bcp.0)
residuals.bcp(bcp.0)
```

---

summary.bcp *Summarizing Bayesian change point analysis results*

Description

Summary and print methods for class bcp.

Usage

```
 ## S3 method for class 'bcp'
summary(object, digits = max(3, .Options$digits - 3), ...)
 ## S3 method for class 'bcp'
print(x, digits = max(3, .Options$digits - 3), ...)
```

Arguments

x	the result of a call to bcp().
object	the result of a call to bcp().
digits	the number of digits displayed in the summary statistics.
...	additional arguments.

Details

The functions print (and return invisibly) the estimated posterior probability of a change point for each position and the estimated posterior means. These results are modeled after the summary method of the coda package (Plummer *et al.*, 2006). If return.mcmc=TRUE (i.e., if full MCMC results are returned), bcp objects can be converted into mcmc objects to view mcmc summaries – see examples below.

Value

The matrix of results is returned invisibly.

Author(s)

Chandra Erdman and John W. Emerson

References

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines (2006), *The coda Package, version 0.10-7*, CRAN: The Comprehensive R Network.

See Also

bcp and plot.bcp.

Examples

```
##### A random sample from a few normal distributions #####
testdata <- c(rnorm(50), rnorm(50, 5, 1), rnorm(50))
bcp.0 <- bcp(testdata)
summary.bcp(bcp.0)
plot.bcp(bcp.0)

##### An MCMC summary from the ``coda'' package #####
if (require("coda")) {
  bcp.0 <- bcp(testdata, return.mcmc=TRUE)
  bcp.mcmc <- as.mcmc(t(bcp.0$mcmc.means))
  summary(bcp.mcmc)
  heidel.diag(bcp.mcmc) # an example convergence diagnostic
                        # from the coda package.
}
```

Index