Package ‘bcp’

August 29, 2016

Type Package
Title Bayesian Analysis of Change Point Problems
Version 4.0.0
Date 2015-07-14
Author Xiaofei Wang, Chandra Erdman, and John W. Emerson
Maintainer John W. Emerson <john.emerson@yale.edu>
Depends graphics, methods, grid
Suggests DNAcopy, coda, strucchange, vegan, ggplot2, igraph
Description Provides an implementation of 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 regression models on a connected graph (Wang and Emerson, 2015); this allows estimation of change point models with multivariate responses. Parallel MCMC, previously available in bcp v.3.0.0, is currently not implemented.
License GPL (>= 2)
Imports Rcpp (>= 0.9.2)
LinkingTo Rcpp, RcppArmadillo
NeedsCompilation yes
Repository CRAN
Date/Publication 2015-07-15 13:23:42

R topics documented:

  bcp-package ................................................................. 2
  bcp ................................................................. 2
  coriell ................................................................. 8
  fitted.bcp ................................................................. 9
  interval.prob ................................................................. 9
  legacyplot ................................................................. 10
  lombard ................................................................. 11
  makeAdjGrid ................................................................. 12
**bcp-package**

*Bayesian Analysis of Change Point Problems*

**Description**

Provides an implementation of the Barry and Hartigan (1993) product partition model for the normal errors change point problem using Markov Chain Monte Carlo. It also (i) extends the methodology to regression models on a connected graph (Wang and Emerson, 2015) and (ii) allows estimation of change point models with multivariate responses. Parallel MCMC, previously available in bcp v.3.0.0, is currently not implemented.

**Author(s)**

Xiaofei Wang, Chandra Erdman, and John W. Emerson

**bcp**

*Performs a Bayesian analysis of change point problems*

**Description**

bcp() implements the Bayesian change point analysis methods given in Wang and Emerson (2015), of which the Barry and Hartigan (1993) product partition model for the normal errors change point problem is a specific case. 1. Multivariate (or univariate) Bayesian change point analysis: We assume there exists an unknown partition of a data series y into blocks such that the mean is constant within each block. In the multivariate case, a common change point structure is assumed; 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. 2. Linear regression Bayesian change point analysis: As with the previous model, we assume the observations (x,y), where x may be multivariate, are partitioned into blocks, and that linear models are appropriate within each block.

If an adjacency structure is provided, the data are assumed to reside on nodes of a graph with the given adjacency structure; additional parameters are used in this graph change point model. If no adjacency structure is provided, the data are assumed to be sequential and the blocks are forced to be contiguous.
Usage

```r
bcp(y, x = NULL, id = NULL, adj = NULL, w0 = NULL, p0 = 0.2, d = 10,
    burnin = 50, mcmc = 500, return.mcmc = FALSE, boundaryType = "node",
    p1 = 1, freqAPP = 20)
```

Arguments

- **y**: a vector or matrix of numerical data (with no missing values). For the multivariate change point problems, each column corresponds to a series.
- **x**: (optional) a matrix of numerical data (with no missing values) representing the predicting variables for a linear regression.
- **id**: (optional) a vector of integers specifying the location ID for each observation in `y`, starting from location 1.
- **adj**: (optional) an adjacency list. Indexing the observations from 1 to `n`, the `i`-th element of the list is a vector of indices (offset by 1) for nodes that share an edge with node `i`.
- **w0**: (optional) a single numeric value in the multivariate case or a vector of values in the regression case; in both, the value(s), between 0 and 1, is/are the parameter(s) in the uniform prior(s) on the signal-to-noise ratio(s). If no value is specified, the default value of 0.2 is used, as recommended by Barry and Hartigan (1993).
- **p0**: (optional) a value between 0 and 1. For sequential data, it is the parameter of the prior on change point probabilities, \( U(0, \ p0) \), on the probability of a change point at each location in the sequence; for data on a graph, it is the parameter in the partition prior, \( p0^l(\rho) \), where \( l(\rho) \) is the boundary length of the partition.
- **d**: (optional) a positive number only used for linear regression change point models. Lower \( d \) means higher chance of fitting the full linear model (instead of the intercept-only model); see prior for \( \tau_S \) in Wang and Emerson (2015).
- **burnin**: the number of burnin iterations.
- **mcmc**: the number of iterations after burnin.
- **return.mcmc**: logical. If set to TRUE, the posterior means and the partitions in each iteration are returned.
- **boundaryType**: (optional) only applicable for graph change point analysis. Values can be “node” (default) if we count nodes in the boundary length calculation, or “edge” if we count edges in the boundary length calculation. See Wang and Emerson (2015) for details.
- **p1**: (optional) only applicable for graph change point analysis. The proportion of Active Pixel Passes run that are the actual Active Pixel Passes specified in Barry and Hartigan (1994). \( p1 = 0 \) corresponds to exclusively using the pseudo-Active Pixel Passes given in Wang and Emerson (2015).
- **freqAPP**: (optional) only applicable for graph change point analysis. A positive integer for the number of Active Pixel Passes run in each step of the MCMC algorithm.
Details

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.

Value

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. In the regression case, the matrix includes posterior means for the response variable.
- `posterior.var` a vector or matrix of the estimated posterior variances. In the regression case, the estimated posterior variances of the response are provided.
- `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.

Author(s)

Xiaofei Wang, Chandra Erdman, and John W. Emerson

References


See Also

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

Examples

```r
# univariate sequential data
set.seed(5)
x <- c(rnorm(50), rnorm(50, 5, 1), rnorm(50))
bcp.1a <- bcp(x)
plot(bcp.1a, main="Univariate Change Point Example")
legacyplot(bcp.1a)

# a hard problem with 1 true change point
set.seed(5)
x <- rep(c(0, 1), each=50)
y <- x + rnorm(50, sd=1)
bcp.1b <- bcp(y)
plot(bcp.1b, main="Univariate Change Point Example")

# multivariate sequential data
set.seed(5)
x <- rnorm(6, sd=3)
y <- rbind(cbind(rnorm(50, x[1]), rnorm(50, x[2]), rnorm(50, x[3])),
          cbind(rnorm(50, x[4]), rnorm(50, x[5]), rnorm(50, x[6])))
bcp.2a <- bcp(y)
plot(bcp.2a, main="Multivariate (k=3) Change Point Example")
plot(bcp.2a, separated=TRUE, main="Multivariate (k=3) Change Point Example")

# a harder problem in k=5 dimensions
set.seed(5)
```
means1 <- rep(0, 5)
means2 <- rep(1, 5)
x <- rbind(matrix(rep(means1, each=50), nrow=50),
           matrix(rep(means2, each=50), nrow=50))
y <- x + rnorm(length(x), sd=1)
bcp.2b <- bcp(cbind(y))
plot(bcp.2b, main=“Multivariate (k=5) Change Point Example”)

##### linear models with sequential data #####
# 1 true change point at location 50; the predicting variable x is not related to location
x <- rnorm(100)
b <- rep(c(3,-3), each=50)
y <- b*x + rnorm(100)
bcp.3a <- bcp(y, x)
# in the two plots that follow, the location IDs are used as the plot characters
par(mfrow=c(1,2))
plot(y ~ x, type="n", main=“Linear Regression: Raw Data”)
text(x, y, as.character(1:100), col=(b/3)+2)
plot(y ~ x, type="n", main=“Linear Regression: Posterior Means”)
text(x, bcp.3aposterior.mean[,1], as.character(1:100), col=(b/3)+2)
plot(bcp.3a, main=“Linear Regression Change Point Example”)

# 1 true change point at location 50; the predicting variable x is equal to location
x <- 1:100
b <- rep(c(3,-3), each=50)
y <- b*x + rnorm(100, sd=50)
bcp.3b <- bcp(y, x)
plot(bcp.3b, main=“Linear Regression Change Point Example”)

##### univariate data on a grid #####
set.seed(5)
adj <- makeAdjGrid(20)
z <- rep(c(0, 2), each=200)
y <- z + rnorm(400, sd=1)
out <- bcp(y, adj=adj, burnin=500, mcmc=500)

if (require(“ggplot2”)) {
  df <- data.frame(mean=z, data = y, post.means = out$posterior.mean[,1],
                   post.probs = out$posterior.prob,
                   i = rep(1:20, each=20), j = rep(1:20, times=20))

  # visualize the means
  g <- ggplot(df, aes(i,j)) +
       geom_tile(aes(fill = mean), color='white') +
       scale_fill_gradientn(limits=range(y), colours=c('white', 'steelblue'))+
       ggtitle("True Means")
  print(g)

  # visualize the data
  g <- ggplot(df, aes(i,j)) +
       geom_tile(aes(fill = data), color='white') +
       scale_fill_gradientn(limits=range(y), colours=c('white', 'steelblue'))+
       ggtitle("Observed Data")
# visualize the posterior means/probs

g <- ggplot(df, aes(i,j)) +
  geom_tile(aes(fill = post.means), color='white') +
  scale_fill_gradientn(limits=range(y), colours=c('white', 'steelblue')) +
  ggttheme("Posterior Means")

print(g)

g <- ggplot(df, aes(i,j)) +
  geom_tile(aes(fill = post.probs), color='white') +
  scale_fill_gradientn(limits=c(0, 1), colours=c('white', 'steelblue')) +
  ggttheme("Posterior Boundary Probabilities")

print(g)

## Not run:

#### multivariate data on a grid ####

sd=3

x <- rnorm(6)
y <- rbind(cbind(rnorm(50, x[1]), rnorm(50, x[2]), rnorm(50, x[3])),
           cbind(rnorm(50, x[4]), rnorm(50, x[5]), rnorm(50, x[6])))
adj <- makeAdjGrid(10)
a <- bcp(y, adj=adj, p0=0.4, burnin=500, mcmc=500)

#### linear models on a grid ####

b <- rep(c(3,-3), each=50)
y <- b*x + rnorm(100)
adj <- makeAdjGrid(10)
a <- bcp(y, x, adj=adj, p0=0.4, burnin=500, mcmc=500)

#### linear models on a grid using pseudo-APPs ####

b <- rep(c(3,-3), each=50)
y <- b*x + rnorm(100)
adj <- makeAdjGrid(10)
a <- bcp(y, x, adj=adj, p0=0.4, burnin=500, mcmc=500, p1 = 0)

## End(Not run)

#### univariate data on a graph ####

## Not run:

demo(bcpgraph)

## End(Not run)

#### Real Data Examples ####

## Not run:
# Coriell chromosome 11: univariate sequential data
demo(coriell)
coriell

Array CGH data set of Coriell cell lines

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

Usage
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/full/ng754.html

References

Examples
demo(coriell)
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()`.
- `...` (optional) additional arguments, ignored.

Value
Fitted values extracted from the bcp object.

Author(s)
Xiaofei Wang, 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)
residuals(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 of sequential observations; 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

For sequential data only, 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)

Xiaofei Wang, Chandra Erdman, and John W. Emerson

See Also

bcp and plot.bcp.

Examples

```r
### 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.0, main="Univariate Change Point Example")
interval.prob(bcp.0, 45, 55)
```
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

```r
#### 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)
legacyplot(bcp.0)
```

lombard

Milling machine indentation data

Description

Radii of 100 circular indentations cut by a milling machine.

Usage

lombard

Format

A vector of length 100 containing the individual radii.

Source


References

makeAdjGrid

Creating the adjacency structure for grid graphs

Description

makeAdjGrid() produces a sparse representation of the adjacency structure for grid graphs, useful as the adj argument in bcp().

Usage

makeAdjGrid(n, m = NULL, k = 8)

Arguments

- **n**: the number of rows of vertices in the graph data.
- **m** (optional): the number of column of vertices in the graph data. If not given, we assume m = n.
- **k** (optional): the number of neighbors assumed for a typical vertex (see details below), either 4 or 8. Default number of neighbors is assumed to be 8.

Details

makeAdjGrid() produces a list representation of the adjacency structure for grid graphs. The i-th entry in the list gives a vector of neighbor ids for the i-th node. Note that neighbor ids are offset by 1 because indexing starts at 0 in C++. If k = 8, then we assume each node is joined via edges to its 8 neighbors in the (top left, top middle, top right, left, right, bottom left, bottom middle, and bottom right) directions, where applicable. If k = 4, then we assume each node is joined via edges to its 4 neighbors in the (top, right, bottom, left) directions, where applicable.

Author(s)

Xiaofei Wang
See Also

bcp for performing Bayesian change point analysis.

Examples

# generates an adjacency list for a 10 node by 5 node grid, assuming a maximum of 8 neighbors
adj <- makeAdjGrid(10, 5)

# generates an adjacency list for a 10 node by 5 node grid, assuming a maximum of 4 neighbors
adj4 <- makeAdjGrid(10, 5, 4)

### show a grid example
## Not run:
set.seed(5)
adj <- makeAdjGrid(20)
z <- rep(c(0, 2), each=200)
y <- z + rnorm(400, sd=1)
out <- bcp(y, adj=adj, burnin=500, mcmc=500)

if(require("ggplot2")) {
  df <- data.frame(mean=z, data = y, post.means = out$posterior.mean[,1],
                   post.probs = out$posterior.prob,
                   i = rep(1:20, each=20), j = rep(1:20, times=20))

  # visualize the data
  g <- ggplot(df, aes(i,j)) +
       geom_tile(aes(fill=data), color='white') +
       scale_fill_gradientn(limits=range(y), colours=c('white', 'steelblue'))+
       ggtitle("Observed Data")
  print(g)

  # visualize the means
  g <- ggplot(df, aes(i,j)) +
       geom_tile(aes(fill=mean), color='white') +
       scale_fill_gradientn(limits=range(y), colours=c('white', 'steelblue'))+
       ggtitle("True Means")
  print(g)

  # visualize the posterior means/probs
  g <- ggplot(df, aes(i,j)) +
       geom_tile(aes(fill=post.means), color='white') +
       scale_fill_gradientn(limits=range(y), colours=c('white', 'steelblue'))+
       ggtitle("Posterior Means")
  print(g)

  g <- ggplot(df, aes(i,j)) +
       geom_tile(aes(fill=post.probs), color='white') +
       scale_fill_gradientn(limits=c(0, 1), colours=c('white', 'steelblue'))+
       ggtitle("Posterior Boundary Probabilities")
  print(g)
}
New Haven Housing

## Description

Location, 2011 assessed value, and physical characteristics for 244 houses in a region of New Haven, CT.

## Usage

NewHavenHousing

## Format

A matrix containing location, 2011 assessed value, and physical characteristics in the columns.

## Source

The data can be scraped from the New Haven, CT Online Assessment Database [http://gis.vgsi.com/newhavenct/](http://gis.vgsi.com/newhavenct/)

## References


## Examples

```r
## Not run:
demo("NewHaven")

## End(Not run)
```
plot.bcp

Plotting Bayesian change point results

Description

plot.bcp() produces summary plots of the results of bcp(). Currently, only the summary plots for serial data are implemented. If an adjacency structure (adj) is provided, then the data are assumed to reside on nodes of a general graph. Additional parameters are used in this graph change point model.

Usage

```r
## 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, xlab = NULL,
  xaxlab = 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` logical. If set to TRUE and the data is multivariate, each series is plotted separately.
- `outer.margins` (optional) list of units specifying the left, bottom, right and top margins. For more information on units, see the documentation for grid.
- `lower.area` (optional) unit specifying the proportion of the plot occupied by the posterior probabilities of change points.
- `size.points` (optional) unit specifying the size of the data points.
- `pch.points` (optional) unit specifying the style of the data points.
- `colors` (optional) vector specifying the colors in which to plot each data series.
- `main` (optional) plot title. Use "" for no title.
- `xlab` (optional) a character string specifying the x-axis label. Defaults to "Location".
- `xaxlab` (optional) a vector having length equal to the number of observations giving the x-axis tick labels. Defaults to the sequence from 1 to n.
- `cex.axes` (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.
- `...` (optional) 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.

Author(s)

Xiaofei Wang, 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.0, main="Multivariate (k=2) Change Point Example")
plot(bcp.0, separated=TRUE, main="Multivariate (k=2) Change Point Example")

Quebec Rivers

Quebec river streamflow data

Description

Annual January to June streamflow amounts (measured in $L/(km^2.s)$) for four rivers in Quebec: Baleine, Churchill Falls, Manicouagan, and Romaine.

Usage

QuebecRivers

Format

A matrix containing streamflow amounts for the years 1972 to 1994.

Source

The data can be obtained from the Centre d’expertise hydrique Quebec. https://www.cehq.gouv.qc.ca/hydrometrie/index-en.htm
References


2. L. Perrault et al. (2000). Retrospective multivariate Bayesian change-point analysis: a simultaneous single change in the mean of several hydrological sequences, Stochastic Environmental Research and Risk Assessment, 14, 243-261.

Examples

data("QuebecRivers")
bcpr.rivers <- bcp(QuebecRivers)
plot(bcpr.rivers, main="Quebec River Streamflow Change Point Analysis",
     xlab="Year", xaxlab = 1972:1994)

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

RealInt

Format

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

Source


References


Examples

demo(RealInt)
residuals.bcp

Extract model residuals

Description

residuals method for class bcp.

Usage

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

Arguments

- `object`: the result of a call to `bcp()`.
- `...`: (optional) additional arguments, ignored.

Value

Residuals extracted from the `bcp` object.

Author(s)

Xiaofei Wang, Chandra Erdman, and John W. Emerson

See Also

- `bcp` and `plot.bcp`

Examples

```r
#### A random sample from a few normal distributions ####
testdata <- c(rnorm(50), rnorm(50, 5, 1), rnorm(50))
bcp.0 <- bcp(testdata)
residuals(bc.0)
```
Summary and print methods for class `bcp`.

### Usage

```r
## 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

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

### 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)

Xiaofei Wang, Chandra Erdman, and John W. Emerson

### See Also

- `bcp` and `plot.bcp`. 
Examples

##### A random sample from a few normal distributions #####
```r
testdata <- c(rnorm(50), rnorm(50, 5, 1), rnorm(50))
bcp.0 <- bcp(testdata)
summary(bc.p.0)
plot(bc.p.0, main="Univariate Change Point Example")
```

##### An MCMC summary from the `coda` package #####
```r
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

*Topic datasets
  bcp, 2
  coriell, 8
  fitted.bcp, 9
  interval.prob, 9
  legacyplot, 10
  lombard, 11
  makeAdjGrid, 12
  NewHavenHousing, 14
  plot.bcp, 15
  QuebecRivers, 16
  RealInt, 17
  residuals.bcp, 18
  summary.bcp, 19

bcp, 2, 10, 11, 13, 16, 18, 19
bcp-package, 2
coriell, 8
fitted.bcp, 9
interval.prob, 9
legacyplot, 4, 10, 16
lombard, 11
makeAdjGrid, 12
NewHavenHousing, 14
plot.bcp, 4, 5, 9–11, 15, 18, 19
print.bcp, 4, 5, 9, 11, 16
print.bcp(summary.bcp), 19
QuebecRivers, 16
RealInt, 17
residuals.bcp, 18
summary.bcp, 4, 5, 9, 11, 16, 19