The R Package Management System: Bayesian Change Point Analysis

John W. Emerson

http://www.stat.yale.edu/~jay/
Associate Professor of Statistics, Yale University
(Professor Emerson prefers to be called “Jay”)

I would like to thank Chandra Erdman (Yale GRD ’08) for her collaboration on the Bayesian change point analysis and package bcp. Chandra is now at the Census Bureau.

Please feel free to ask questions along the way!

http://www.stat.yale.edu/~jay/Brazil/Campinas/bcpANDpackages/
http://www.stat.yale.edu/~jay/RPC
Outline

1. Overview
2. Bayesian Change Point Analysis
3. The R Package Management System
4. The C/C++ Interface
5. Parallel Programming
Why R?

R is the *lingua franca* of statistics:

- The syntax is simple and well-suited for data exploration and analysis.
- It has excellent graphical capabilities.
- It is extensible, with over 2500 packages available on CRAN alone.
- It is open source and freely available for Windows/MacOS/Linux platforms.

This talk emphasizes the importance of the package management system. Much of the success of R should be attributed to:

- Ross & Robert’s early decision to go open-source and encourage collaboration, and
- the growth of CRAN and the success of the package management system.
Example: Coriell cell lines (raw data)
Example: Coriell cell lines (bcp analysis)

See http://cran.r-project.org/web/packages/bcp/.

> library(bcp)
> data(coriell)
>
> chrom11 <- as.vector(na.omit(  
+ coriell$Coriell.05296[
+ coriell$Chromosome == 11]))
>
> bcp.11 <- bcp(chrom11)
> plot(bcp.11)
Example: Coriell cell lines (bcp output)
The Bayesian Change Point Analysis

Overview

The R Package Management System

The C/C++ Interface

Parallel Programming

The Bayesian change point model

\( \rho \) unknown partition into continuous blocks, with the transition between blocks being the “change points.”

\( \rho \) probability of a change point at position \( i \), independently for all \( i \).

\( X_i \) observations assumed independent \( N(\mu_i, \sigma^2) \), where in this notation the \( \mu_i \) are equal for all \( i \) within a block.

\( \mu_{jk} \) mean of block from position \( j + 1 \) to \( k \), with prior \( N(\mu_0, \sigma^2_0/(k - j)) \).

Note: larger deviations from \( \mu_0 \) are expected for shorter blocks, but weak signals can be detected when sufficient data are available.

\( w \) defined as \( \sigma^2/(\sigma^2 + \sigma^2_0) \) for convenience.

Possible point of notational confusion: conditional on \( \rho \) (the partition into blocks), \( \mu_i \equiv \mu_{jk} \) for all \( i \) in block \( jk \).
The Bayesian change point priors

\[ \pi(\mu_0) = 1, \quad -\infty \leq \mu_0 \leq \infty \]

\[ \pi(\sigma^2) = \frac{1}{\sigma^2}, \quad 0 \leq \sigma^2 \leq \infty \]

\[ \pi(p) = \frac{1}{p_0}, \quad 0 \leq p \leq p_0 \]

\[ \pi(w) = \frac{1}{w_0}, \quad 0 \leq w \leq w_0 \]

\[ \pi(\rho) = \frac{1}{p_0} \left[ \int_0^{p_0} p^{b-1} (1 - p)^{n-b} \, dp \right] \]

where \( p_0 \) and \( w_0 \) are pre-selected in \([0, 1]\) (values of 0.2 are the default in package \texttt{bcp} and work well in a wide range of cases), and \( b \) is the number of blocks in \( \rho \).
Notes on Bayesian change point analysis

- An exact implementation of the Bayes procedure is possible but the calculations would be $O(n^3)$.
- Package **bcp** provides a fast $O(n)$ MCMC implementation:
  - inefficient MCMC would be $O(n^2)$
  - solves some nasty numerical problems with large data
  - supports parallel MCMC
  - currently being extended for multivariate series with a common change point structure
Second example: simulated aberrations of length 2, 5, 10, 20, and 40
If you want to build R packages, you’ll need the full R build environment (not just the pre-compiled R binary that most of us use from CRAN). See

http://www.stat.yale.edu/~jay/RPC/RPackages.pdf
A simple package: function `babywhatis()`

```r
> ls()
[1] "bcp.11"  "chrom11"  "coriell"
> rm(list = ls())
> ls()
character(0)

> babywhatis <- function(x) {
+   if (!is.data.frame(x)) {
+     x <- data.frame(x)
+     warning("Object coerced to a data frame.\n")
+   }
+   return(unlist(lapply(x, class)))
+ }
> ls()
[1] "babywhatis"
```
A simple package: the package skeleton

> package.skeleton("MyToolkit")

Creating directories ...
Creating DESCRIPTION ...
Creating Read-and-delete-me ...
Saving functions and data ...
Making help files ...
Done.
Further steps are described in
  './MyToolkit/Read-and-delete-me'.

Let’s go investigate together; we’ll explore the package structure, make minor modifications, and will check/build/install it. More information is available in http://www.stat.yale.edu/~jay/RPC/RPackages.pdf
At this point, we’ll just glance at package \texttt{bcp} quickly. I want to introduce the C/C++ interface with a simple example before looking more closely at \texttt{bcp}.
A simple example: column minima

This material wouldn’t display well in slides. Again see

http://www.stat.yale.edu/~jay/RPC/RPackages.pdf

and, specifically, materials in

http://www.stat.yale.edu/~jay/RPC/MyToolkitWithC/
Package **bcp** uses the `.C()` interface instead of `.Call()`, though I’d like to change this in a new version.
The user may register any one of several “parallel backends” like `doMC` or `doSNOW`, or none at all. The code will either run sequentially or will make use of the parallel backend, if specified, without code modification.

```r
> library(foreach)
> library(doMC)
> registerDoMC(2)

> a <- 10
> ans <- foreach(i = 1:5, .combine = c) %dopar% +
>     a + i^2
> ans

[1]  11  14  19  26  35
```
An older version of bcp used NetWorkSpaces for parallel MCMC; this was very difficult to install and use, and the code was not portable to other parallel environments.

The new bcp uses Steve Weston’s foreach package, and the user may choose from a variety of parallel backends.

I strongly recommend foreach() for parallel programming, to both users and package developers.

Again, see supplementary materials for more information.
Thanks

- Bell Laboratories (Rick Becker, John Chambers and Allan Wilks), for development of the S language
- Ross Ihaka and Robert Gentleman, for their work and unselfish vision for R
- The R Core team
- John Hartigan, for years of teaching and mentoring
- John Emerson (my father, Middlebury College), for getting me started in statistics and computer science
- All my students (and Chandra Erdman in particular on this project) for their willingness argue with me