Overview of CNPBayes package

Jacob Carey, Steven Cristiano, and Robert Scharpf

May 5, 2017

Contents

1 Introduction

CNPBayes models multi-modal densities via a hierarchical Bayesian Gaussian mixture model. The major application of this model is the estimation of copy number at copy number polymorphic loci (CNPs). Two versions of the mixture model are implemented. A *standard* model, referred to as a *marginal* model, that has one mean and standard deviation for each component, and a *batch* model with batch-specific means and standard deviations. Approximation of the posterior is by Markov Chain Monte Carlo (MCMC) written in C++ using the Rcpp package (Eddelbuettel and François 2011).

For an EM-implementation of Gaussian mixture models for CNPs, see the Bioconductor package CNVtools (Barnes et al. 2008). A Bayesian extension of this model by some of the same authors was developed to automate the analysis of the Welcome Trust Case Control Consortium (WTCCC) genotype data (Cardin et al. 2011) and implemented in the R package CNVCALL (http://niallcardin.com/CNVCALL).

This vignette provides a concise workflow for fitting mixture models in large array-based genome-wide association studies. We refer the reader to other vignettes included with this package for details regarding implementation.

```
library(CNPBayes)
library(GenomicRanges)
```

2 Workflow

2.1 Delineate CNPs for each ancestry group

Provided in the `CNPBayes` package is example `SnpArrayExperiment` and `GRangesList` data.

```
se <- readRDS(system.file("extdata", "simulated_se.rds", package="CNPBayes"))
grl <- readRDS(system.file("extdata", "grl_deletions.rds", package="CNPBayes"))
```

Using this data, we identify CNP loci and summarize within sample and locus by median.

```
cnv.region <- consensusCNP(grl, max.width=5e6)
## Warning: In BioC 3.5, the 'force' argument was replaced by the more
##   flexible 'pruning.mode' argument, and is deprecated. See ?seqinfo
##   for the supported pruning modes. Note that 'force=TRUE' is
##   equivalent to 'pruning.mode="coarse"'.
i <- subjectHits(findOverlaps(cnv.region, rowRanges(se)))
med.summary <- matrixStats::colMedians(assays(se)[["cn"]][i, ], na.rm=TRUE)
```

See Identifying Copy Number Polymorphisms for instructions on finding CNPs with a `SnpArrayExperiment` and `GRangesList`.

A `MixtureModel` is constructed using this summarized data for a CNP locus with a call to `MarginalModel` or `BatchModel`, depending on whether the simulation should be *marginal* across batch effect, or *hierarchical*. The batch of a subject is determined by the chemistry plate and is specified using the `batch` parameter in the `BatchModel`.

```
model.marginal <- MarginalModel(data=med.summary)
model.batch <- BatchModel(data=med.summary,
                          batch=c(rep(1, 12), rep(2, 23)))
```

For details about model construction and other optional parameters, see Bayesian mixture models for copy number estimation.

2.2 Fitting mixture models at each CNP

Independently for each CNP, we group the samples into batches using the `collapseBatch` function and create a new model.

```
model.batch <- BatchModel(data=med.summary,
                          batch=collapseBatch(model.batch))
```

For array-based estimates and germline genomes, there are typically between 1 and 4 copy number states at any given CNP. Since we do not know the number of components a priori, we fit a model for each k. In addition, we fit models with and without a term for batch.

The `posteriorSimulation` function takes a constructed `MixtureModel` and `k`, a vector of the number of components for which to fit models. The posterior simulations of each model is held in two lists - one for `MarginalModel`'s and one for `BatchModel`'s.

```
set.seed(1337)
mlist.marginal <- posteriorSimulation(model.marginal, k=1:4)
mlist.batch <- posteriorSimulation(model.batch, k=1:4)
```

2.3 Selecting a model

Marginal likelihood is estimated for each of the 8 models and the Bayes' factors is used to select one of these models for further study.

```
# marginal likelihood of each model
marginal.lik <- marginalLikelihood(mlist.marginal)
batch.lik <- marginalLikelihood(mlist.batch)
## Warning in FUN(X[[i]], ...): The model for k=3 may be overfit. This can
## lead to an incorrect marginal likelihood
## Warning in FUN(X[[i]], ...): The model for k=4 may be overfit. This can
## lead to an incorrect marginal likelihood

# bayes factor for comparing top two models
logBayesFactor(marginal.lik)
##         SB1         SB2         SB3         SB4
## SB1  0.00000 -51.741343 -45.663291 -39.189282
## SB2 51.74134   0.000000   6.078053  12.552061
## SB3 45.66329  -6.078053   0.000000   6.474008
## SB4 39.18928 -12.552061  -6.474008   0.000000
logBayesFactor(batch.lik)
##          MB1       MB2 MB3 MB4
## MB1  0.00000 -24.84295  NA  NA
## MB2 24.84295   0.00000  NA  NA
## MB3       NA        NA  NA  NA
## MB4       NA        NA  NA  NA
```

2.4 Extracting MAP estimates and posterior probabilities

For the selected model, we can view the *maximum a posteriori* estimates for component.

```
selected.model <- mlist.marginal[[which.max(marginal.lik)]]
map(selected.model)
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
```

References

Barnes, Chris, Vincent Plagnol, Tomas Fitzgerald, Richard Redon, Jonathan Marchini, David Clayton, and Matthew E Hurles. 2008. "A Robust Statistical Method for Case-Control Association Testing with Copy Number Variation." *Nat Genet* 40 (10). Wellcome Trust Sanger Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge, UK.: 1245–52. doi:10.1038/ng.206.

Cardin, Niall, Chris Holmes, Peter Donnelly, and Jonathan Marchini. 2011. "Bayesian Hierarchical Mixture Modeling to Assign Copy Number from a Targeted Cnv Array." *Genet. Epidemiol.* doi:10.1002/gepi.20604.

Eddelbuettel, Dirk, and Romain François. 2011. "Rcpp: Seamless R and C++ Integration." *Journal of Statistical Software* 40 (8): 1–18. http://www.jstatsoft.org/v40/i08/.