Binomial Mixtures

Section 6.1 of Drton and Plummer (2017) considers singular BIC calculations for binomial mixture models. This section uses simulated data. We therefore introduce a convenience function for simulating from a binomial mixture model.

```
rBinomMix = function(n, alpha, theta, size) {
  nindex = rmultinom(1, size=n, prob=alpha)
  rbinom(n, size, rep(theta, nindex))
}
```

The `rBinomMix()` function generates `n` samples from a mixture model in which component `i` has a binomial distribution with probability parameter `theta[i]` and size parameter `size` (assumed to be common to all mixture components). The prior probability of component `i` is proportional to `alpha[i]`. Model singularities arise when two mixture components h and l have the same probability parameters $\theta_h = \theta_l$ or a mixture weight is zero.

We now set the parameters for the true, data-generating distribution. We choose a 4-component model in which each component has equal prior weight and the probability parameters range between 0.2 and 0.8. The number of trials is 30 for all mixture components.

```
numTrials = 30
trueNumComponents = 4
alpha = rep(1 / trueNumComponents, trueNumComponents)
theta = seq(0.2, 0.8, length = trueNumComponents)
```

Figure 1 shows a histogram of 10000 random samples from this mixture model.

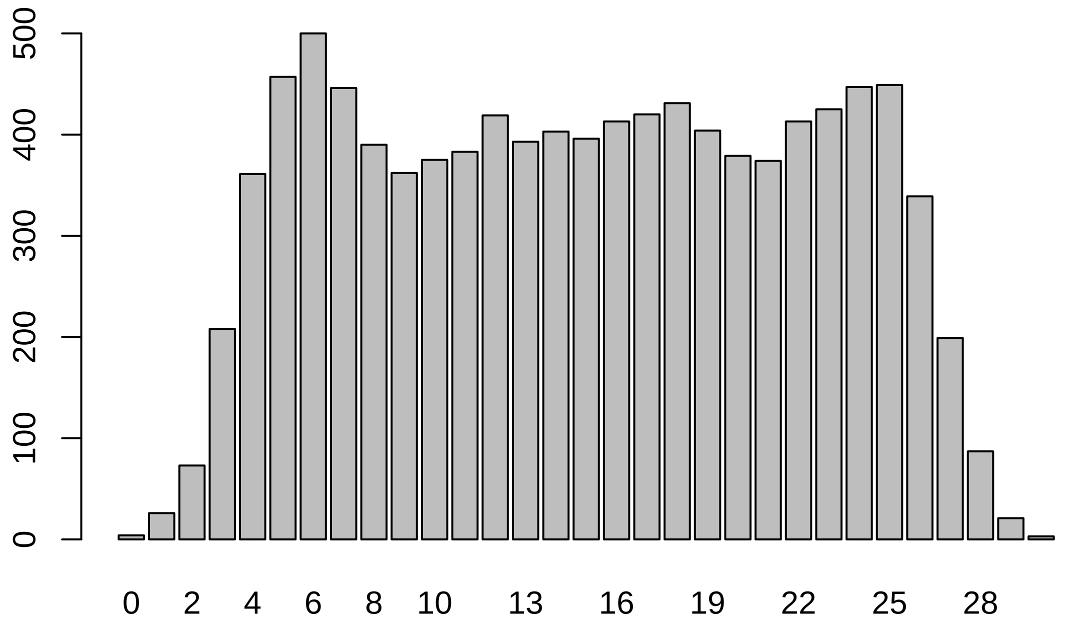

Figure 1: 10000 random samples from a binomial mixture model with 4 components

The variable `maxNumComponents` sets the maximum number of components considered in the calculation of BIC and sBIC.

```
maxNumComponents = 7
```

To illustrate the singular BIC for binomial mixture models we step through the calculations for a single simulated data set of 100 observations.

```
set.seed(11)
Y = rBinomMix(100, alpha, theta, numTrials)
library(sBIC)
binomMix = BinomialMixtures(maxNumComponents, phi = 1)
```

This creates an object `binomMix` of class "BinomialMixtures" containing the approximate learning coefficients for mixture models with up to 7 components. Learning coefficients for binomial mixtures are not fully characterized, but an upper bound is given by Drton and Plummer (2017, equation 6.7). The "BinomialMixtures" class uses these bounds as proxies for the learning coefficients if the penalty parameter `phi` is set to `phi=1`. The extension to general Dirichlet hyperparameter `phi` is given in Drton and Plummer (2017, equation 6.15).

For binomial mixtures the `sBIC` function takes as its first argument a 2-column integer matrix where the first column gives the number of success and the second gives the number of failures.

```
a = sBIC(cbind(Y, numTrials-Y), binomMix)
```

The models represented by `binomMix` are fitted by functions in the `flexmix` package (Gruen and Leisch 2008).

Note that the object `binomMix` is modified by this call to `sBIC`. The "BinomialMixtures" class inherits from the "Object" class in the "R.methodsS3" package which uses call-by-reference semantics. On exit from the `sBIC()` function, `binomMix` contains a copy of the supplied data as well as maximum likelihood values for all fitted models.

The BIC and sBIC values for these data are most easily compared by standardizing them so that the maximum value is zero.

```
a$BIC - max(a$BIC)
```

```
## [1] -263.591396  -10.480096    0.000000   -2.339031   -6.944908  -11.549635
## [7]  -16.155156
```

```
a$sBIC - max(a$sBIC)
```

```
## [1] -263.5916492  -10.4803489    0.0000000   -0.4634676   -2.8027388
## [6]   -5.1084456   -7.4117377
```

When the prior on the mixture weights is a uniform distribution, alternative, tighter bounds on the learning coefficients can be derived from Rousseau and Mengersen (2011). These bounds are implemented by the "BinomialMixtures" class with penalty `phi=0.5`. The penalty parameter can be reset by using the `setPhi` member function.

```
binomMix$setPhi(0.5)
```

The singular BIC can then be recalculated. It is not necessary to resupply the data to the `sBIC` function since, as noted above, it already contains the data and maximum likelihood estimates. Supplying `NULL` to the `sBIC()` function will recalculate the singular BIC without refitting the models

```
a = sBIC(NULL, binomMix)
a$sBIC - max(a$sBIC)
```

```
## [1] -263.8937369  -10.7824367   -0.3014814    0.0000000   -1.2975122
## [6]   -2.4887688   -3.6527034
```

To empirically investigate the asymptotic properties of sBIC for binomial models, we consider increasing sample sizes of 50, 200, and 500 (represented by the variable `sampleSizes`) with 200 simulations for each sample size (represented by `nSim`)

```
nSim = 200
sampleSizes = c(50, 200, 500)
set.seed(11)
```

The code below reproduces Figure 6 of Drton and Plummer (2017), which shows frequencies of the estimated number of mixture components (i.e. the model with the highest BIC or sBIC) for increasing sample sizes. The simulations are too slow to be automatically run when this vignette is built, but the interested reader run them manually.

```
bictable <- vector("list", length(sampleSizes))
for (i in  seq_along(sampleSizes)) {
  results <- matrix(0, nrow=nSim, ncol=3, dimnames=list(NULL, c("BIC","sBIC1","sBIC05")))
  for (j in 1:nSim) {
    Y = rBinomMix(sampleSizes[i], alpha, theta, numTrials)
    X = cbind(Y, numTrials - Y)

    binomMix = BinomialMixtures(maxNumComponents, phi = 1)
    out = sBIC(X, binomMix)
    nc.BIC =  which.max(out$BIC)
    nc.sBIC1 = which.max(out$sBIC)

    binomMix$setPhi(0.5)
    out = sBIC(NULL, binomMix)
    nc.sBIC05 = which.max(out$sBIC)
    results[j, ] <- c(nc.BIC, nc.sBIC1, nc.sBIC05)
  }
  bictable[[i]] <- apply(results, 2, tabulate, nbins=maxNumComponents)
}
```

The code below plots the frequency tables produced by the simulations.

```
par(mfrow=c(1, length(bictable)))
for (i in 1:length(bictable)) {
  tab <- t(bictable[[i]])
  barplot(t(bictable[[i]]), beside=TRUE, ylim=c(0, nSim), names.arg=1:maxNumComponents,
          col=c("white","grey","black"),
          legend = c(expression(BIC), expression(bar(sBIC)[1]), expression(bar(sBIC)[0.5])),
          sub=paste("n =", sampleSizes[i])
}
```

Bibliography

- Drton M. and Plummer M. (2017), A Bayesian information criterion for singular models. J. R. Statist. Soc. B; 79: 1-38.
- Gruen B and Leisch F. (2008) FlexMix Version 2: Finite mixtures with concomitant variables and varying and constant parameters Journal of Statistical Software, 28(4), 1-35. URL http://www.jstatsoft.org/v28/i04/
- Rousseau, J. and Mengersen, K. (2011) Asymptotic behaviour of the posterior distribution in overfitted mixture models. J. R. Statist. Soc. B: 73; 689-710.