finite_mixture.R

wnarifin

Thu Aug 9 10:10:15 2018

```
# R in Medicine Meetup
# 20180809
# Wan Nor Arifin
```

Finite Mixture Model: The Bayesian Approach

```
# A good introduction can be found here:
# http://www.mas.ncl.ac.uk/~nmf16/teaching/mas8391/slides8303-4.pdf
# by Malcolm Farrow
```

About

```
# - Let say we have a continuous variable e.g. height that comes from
#   two groups of people. And we have no data on the grouping. Can we extract
#   the mean height for each groups?
# - Finite mixture model (FMM) is a statistical modeling approach to do that.
# - Usually applied to continuous data.
# - In medicine, it can be used to come up with subgroups of disease.
#   e.g. severity of depression etc.
# - Latent class analysis (LCA) is the statistical analog of FMM for categorical
#   data.
#
# We are going to try out FMM using the Bayesian approach
```

Library

```
library(rjags)  # must also install jags in your computer before rjags
```

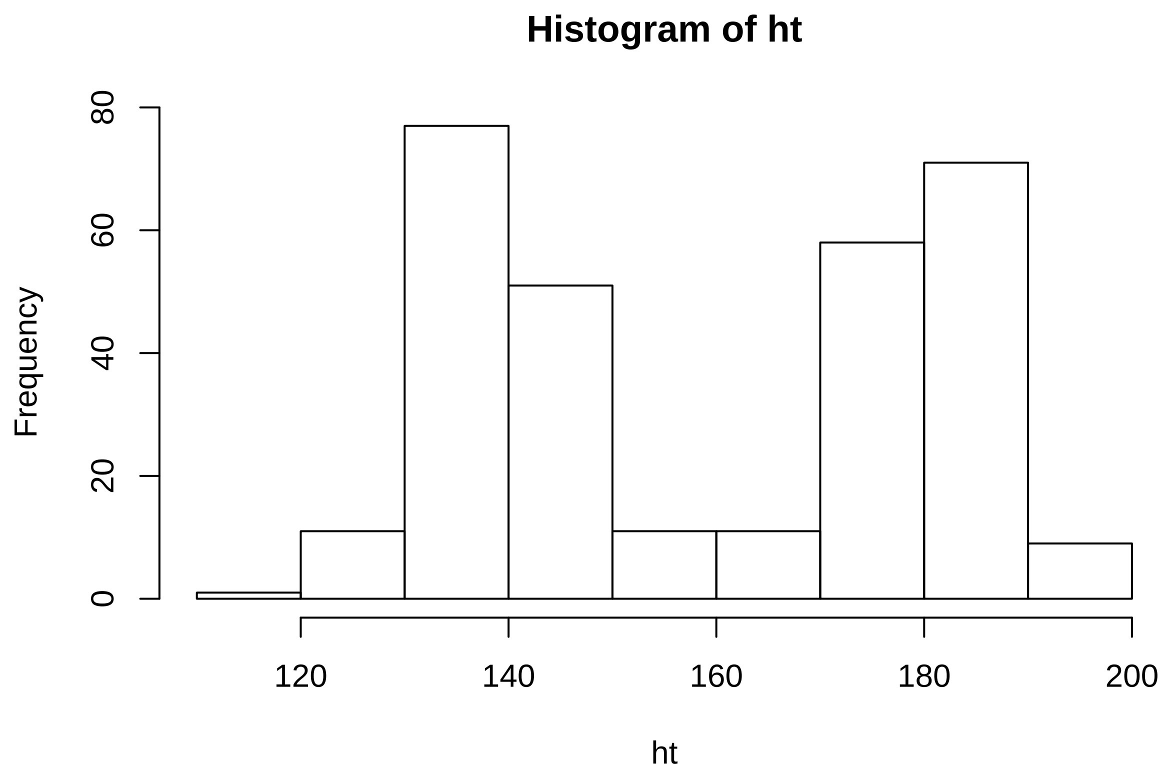
```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.0
```

```
## Loaded modules: basemod,bugs
```

Data

```
# Generate data, say height in cm
# - male = N(mean = 180, SD = 7)
# - female = N(mean = 140, SD = 7)
ht = c(rnorm(150, 180, 7), rnorm(150, 140, 7))
hist(ht)
```



```
jagsdata = list(y = ht, n = length(ht))  # jagsdata must be in list
jagsdata
```

```
## $y
##   [1] 179.8449 172.6949 179.1005 164.9676 170.3837 180.3470 176.8256
##   [8] 174.8983 184.6355 176.1804 181.6680 173.7942 169.0346 179.0531
##  [15] 175.5455 176.9806 187.6085 183.2953 177.8038 175.2606 187.0729
##  [22] 182.3905 175.2466 169.2977 159.6475 190.8427 185.7382 175.8344
##  [29] 183.3111 184.6864 184.2608 180.6597 170.4962 171.6324 167.0174
##  [36] 185.1053 182.1087 181.9271 176.4345 183.7665 182.9442 179.1738
##  [43] 188.1424 188.0440 176.5463 185.2705 180.6770 182.6046 186.8669
##  [50] 177.5195 199.6051 169.6119 194.1590 169.4741 180.4009 176.9088
##  [57] 180.1135 173.1884 174.7010 184.9744 175.7500 184.7439 198.8968
##  [64] 179.1421 183.0035 168.9013 177.9973 182.0890 185.6035 165.6537
##  [71] 174.1230 172.1429 188.5686 180.6573 179.2596 176.2125 181.2054
##  [78] 181.6904 181.1684 183.3992 183.1543 166.3845 190.1326 184.7625
##  [85] 188.8418 183.1879 176.4872 178.5362 180.0691 181.6256 176.8630
##  [92] 191.2564 178.1260 175.7448 175.0336 176.6351 181.8519 175.1643
##  [99] 184.9359 185.1901 183.0032 187.5386 178.8756 183.9849 186.7164
## [106] 182.1510 164.5581 173.5292 183.3976 197.0895 175.6690 186.7526
## [113] 180.7782 180.9287 189.0329 177.7760 181.6542 183.3106 185.1689
## [120] 178.8160 187.4528 174.3306 185.6943 180.4989 175.7321 175.8497
## [127] 181.1113 189.2236 177.9959 176.8358 171.2048 188.1612 178.0217
## [134] 173.0579 167.6912 177.5408 187.3861 193.0378 173.8431 176.3199
## [141] 176.6790 182.3952 179.5263 184.0135 194.7868 187.4374 182.5235
## [148] 171.6397 185.8390 180.2998 143.4311 138.7291 122.1837 134.7767
## [155] 137.7703 131.8847 138.5211 130.3483 127.4873 137.1811 132.2182
## [162] 138.1893 134.4986 152.5561 143.0793 151.1530 134.8579 140.6415
## [169] 147.3362 129.2344 131.2724 140.8601 144.9851 143.8616 138.3739
## [176] 143.2832 136.2646 146.1811 137.7381 131.7659 152.3663 148.1446
## [183] 138.8517 133.7961 139.4732 136.3476 143.3979 138.8593 137.9259
## [190] 137.9685 134.4626 149.0465 145.3434 136.4027 135.0238 139.6472
## [197] 151.3278 143.4917 128.8276 139.7479 146.8975 137.3037 141.2022
## [204] 128.9717 139.8015 144.1727 133.6651 138.9820 139.8963 139.3461
## [211] 148.2199 132.4816 134.0117 148.6602 140.6334 144.7744 151.5987
## [218] 148.6726 148.0771 128.3368 137.0635 142.3638 138.7070 127.9807
## [225] 139.3780 139.7793 151.9599 139.1635 141.8401 142.5159 131.1203
## [232] 130.0829 148.5113 135.9451 150.8073 141.6860 137.6019 139.6271
## [239] 134.7153 142.6607 137.2131 132.3797 143.3461 134.6386 133.2856
## [246] 143.6106 138.6575 125.9800 143.4022 124.0357 138.7758 142.6125
## [253] 133.1696 135.3305 136.3679 143.9968 123.6787 135.1197 139.4076
## [260] 132.4813 130.4643 148.2047 137.0107 146.3849 135.0708 142.6891
## [267] 139.1189 133.2071 138.6552 146.7532 147.1714 131.7336 144.9760
## [274] 143.1433 132.4744 143.4407 145.5470 137.2831 146.4128 133.8697
## [281] 136.7418 140.9999 138.4975 151.7794 141.3807 139.9272 131.7738
## [288] 155.2137 158.1505 133.8444 137.9433 147.6329 142.2511 142.9464
## [295] 119.8460 140.6605 127.5423 144.7574 136.5837 133.4928
##
## $n
## [1] 300
```

FMM

Finite mixture model,

$$p(\boldsymbol{y}_i|\boldsymbol{\vartheta}) = \sum_{k=1}^{K} \eta_k p(\boldsymbol{y}_i|\boldsymbol{\theta}_k)$$

where $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K$ = component parameters, $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_K)$, $\boldsymbol{\vartheta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K, \boldsymbol{\eta})$ = mixture model parameters.

```
# Aim of FMM: Can we "reverse engineer" the process? i.e we obtain the male/female grouping?
#
# Suppose we assume:
# 1. two groups, k = 2
# 2. prior mean = 160, SD = 7, var = 7^2 = 49; precision, tau = 1/var = 1/49
#    mu ~ N(160, 49); in jags mu ~ N(mean, precision) ~ N(160, 1/49)
# 3. prior gamma, mean precision = 1/49,
#    shape, a = 3; rate, b = a/mean = 3/(1/49) = 147
#    tau ~ gamma(3, 147)
# 4. prior beta, pi ~ beta(3, 3) -- gives 50:50 ratio between the groups
```

JAGS code

```
writeLines("
model {
  # data
  for (i in 1:n) {
    c[i] ~ dcat(p[1:2])
    y[i] ~ dnorm(mu[c[i]], tau[c[i]])
  }

  # tau
  for (j in  1:2) {
    tau[j] ~ dgamma(3, 147) # precision, tau
    sigma[j] = sqrt(1/tau[j]) # transform back to SD
  }

  # mu
  mu0 ~ dnorm(160, 1/50) # hierarchical mean, mu
  for (j in 1:2) {
    mu_mu0[j] ~ dnorm(mu0, 1/50) # sample mu
  }
  mu[1:2] = sort(mu_mu0)  # set order constraint, mu1 < mu2

  # pi
  pi ~ dbeta(3, 3)
  p[1] = pi # weight for grp 1
  p[2] = 1 - pi # weight for grp 2
}
", con = "finmix.bug")
```

Fit JAGS model

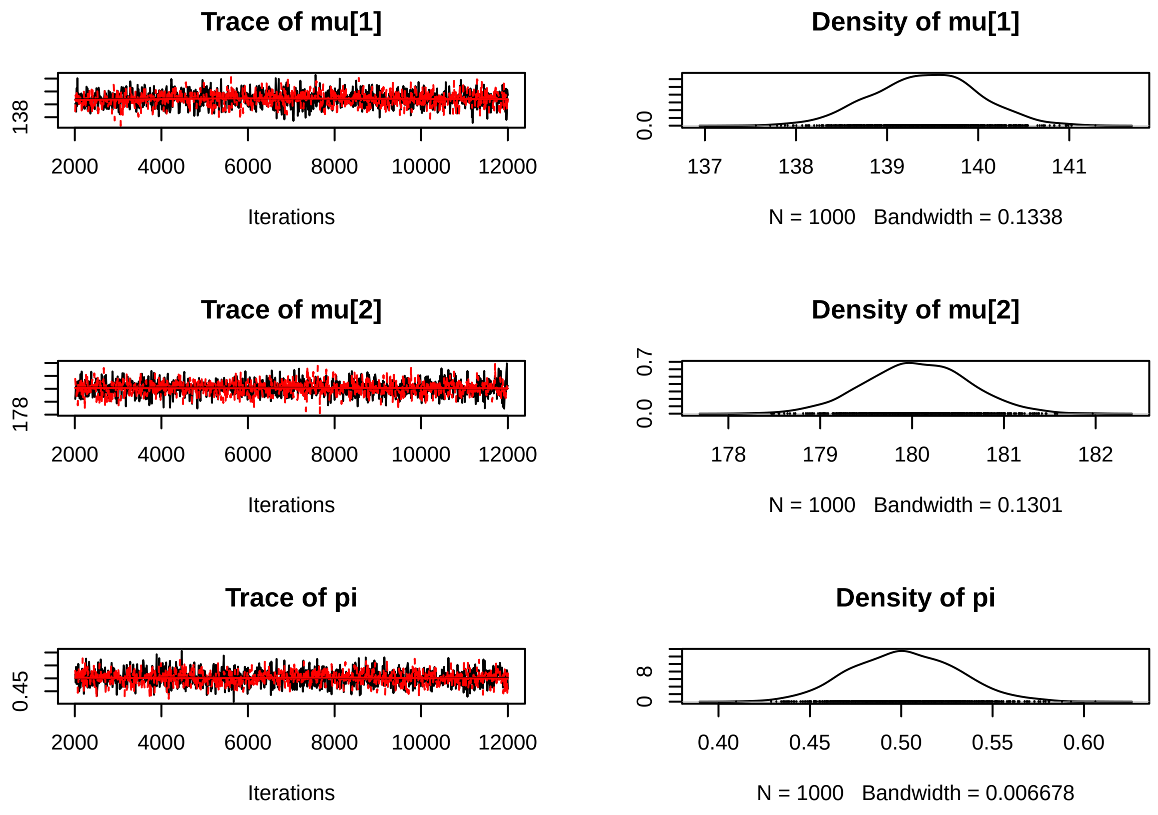
```
jagsmodel = jags.model("finmix.bug", jagsdata, n.chains = 2)
```

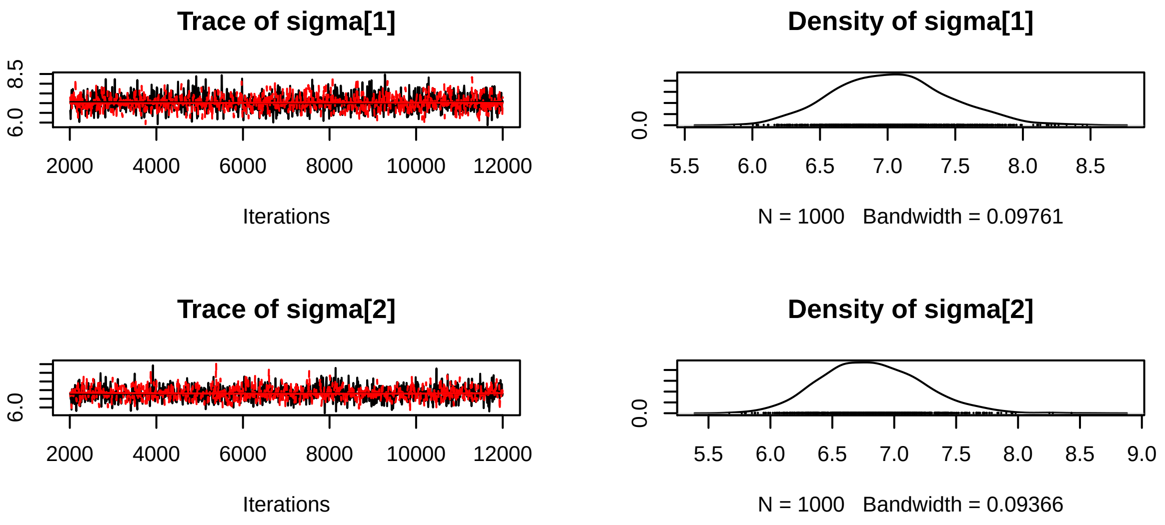
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 300
##    Unobserved stochastic nodes: 306
##    Total graph size: 1223
##
## Initializing model
```

```
update(jagsmodel, 1000)  # burnin = 1000
jagspars = c("mu", "sigma", "pi")
jagssamples = coda.samples(jagsmodel, jagspars, n.iter = 10000, thin = 10)
summary(jagssamples)
```

```
##
## Iterations = 2010:12000
## Thinning interval = 10
## Number of chains = 2
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean      SD  Naive SE Time-series SE
## mu[1]    139.4229 0.58269 0.0130294      0.0130326
## mu[2]     180.0570 0.56480 0.0126292      0.0121657
## pi          0.5017 0.02881 0.0006442      0.0006856
## sigma[1]    7.0340 0.42238 0.0094447      0.0094454
## sigma[2]    6.8180 0.40407 0.0090353      0.0090374
##
## 2. Quantiles for each variable:
##
##               2.5%      25%      50%      75%    97.5%
## mu[1]     138.2798 139.0433 139.4348 139.8167 140.5270
## mu[2]     178.9370 179.6846 180.0482 180.4368 181.1681
## pi          0.4467   0.4814   0.5013   0.5217   0.5597
## sigma[1]    6.2527   6.7343   7.0203   7.2986   7.8987
## sigma[2]    6.0886   6.5381   6.8026   7.0885   7.6687
```

```r
plot(jagssamples)
```




Trace of sigma[1]
6.0
8.5
2000
4000
6000
8000
10000
12000
Iterations
Density of sigma[1]
0.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
N = 1000   Bandwidth = 0.09761
Trace of sigma[2]
6.0
2000
4000
6000
8000
10000
12000
Iterations
Density of sigma[2]
0.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
9.0
N = 1000   Bandwidth = 0.09366