Running a simple simulation

Jack Kuipers, March 20, 2015

For this example we simulate data from a DAG with $n = 5$ nodes and score with the BGe score. This code creates the graphs in the main paper.

```
n <- 5
```

First we load the `R` files needed to run the MCMC codes

```
source("../edgerevandstructure/structurefns.R")
source("../edgerevandstructure/structureMCMC.R")
source("../edgerevandstructure/newedgerevfns.R")
source("../edgerevandstructure/newedgerevmove.R")
source("../orderandpartition/orderMCMC.R")
source("../orderandpartition/orderfns.R")
source("../orderandpartition/partitionMCMC.R")
source("../orderandpartition/partitionmoves.R")
source("../orderandpartition/partitionfns.R")
source("../orderandpartition/samplefns.R")
source("../scoring/combinations.R")
source("../scoring/scorefns.R")
source("../scoring/scoretables.R")
# load the BGe score
source("../scoring/bgescorestable.R")
```

And then generate the data and load the score parameters

```
source("../initialisation/data5nodes.R")
source("../initialisation/scoreparas.R")
```

We can also calculate the score of the DAG the data generated from

```
realDAGlogscore <- sum(DAGnodescore(incidence, n, c(1:n)))
```

Then we build the score table of all parent sets

```
maxparents <- n - 1  # Maximum number of parents to allow
# Fill up a matrix with possible parents
parenttable <- listpossibleparents(maxparents, c(1:n))
tablelength <- nrow(parenttable[[1]])  # size of the table
# Now need to score them!
scoretable <- scorepossibleparents(parenttable, n)
```

so we can also calculate the scores of all orders that DAG is compatible with

```
# calculate score of orders compatible with the DAG that the
# data is generated from
realorderlogscores <- rep(0, length(realorderpermys))
for (j in 1:length(realorderpermys)) {
    realorderscores <- orderscore(n, c(1:n), parenttable, scoretable,
        realorderpermys[[j]])
    # log total score of all DAGs in the order
    realorderlogscores[j] <- sum(realorderscores$totscores)
}
```

and the partition it belongs to

```
realposy <- parttolist(n, realparty)
realpartitionscores <- partitionscore(n, c(1:n), parenttable, scoretable,
    realpermy, realparty, realposy)
# log total score of all DAGs in the partition
realpartitionlogscore <- sum(realpartitionscores$totscores)
```

We will also perform all the runs with the same seed

```
seednumber <- 101  # seed number
```

1 Structure MCMC

For structure MCMC we run for 5000 iterations

```
iterations <- 50000  #number of iterations in the chain
stepsave <- iterations/1000  #how often to save the result
```

and first we allow standard edge reversal by setting

```
revallowed <- 1  # allow standard edge reversals
```

and exclude the new edge reversal move by giving the `moveprobs` vector a single element

```
moveprobs <- c(1)  # having length 1 disallows the new edge reversal move
```

Starting with a DAG with no edges

```
startDAG <- matrix(numeric(n * n), nrow = n)  # starting DAG is empty say
```

we can then run structure MCMC as follows

```
set.seed(seednumber)  # set the seed
example <- structureMCMC(n, startDAG, iterations, stepsave, maxparents,
    parenttable, scoretable, revallowed, moveprobs)  # run the MCMC code
```

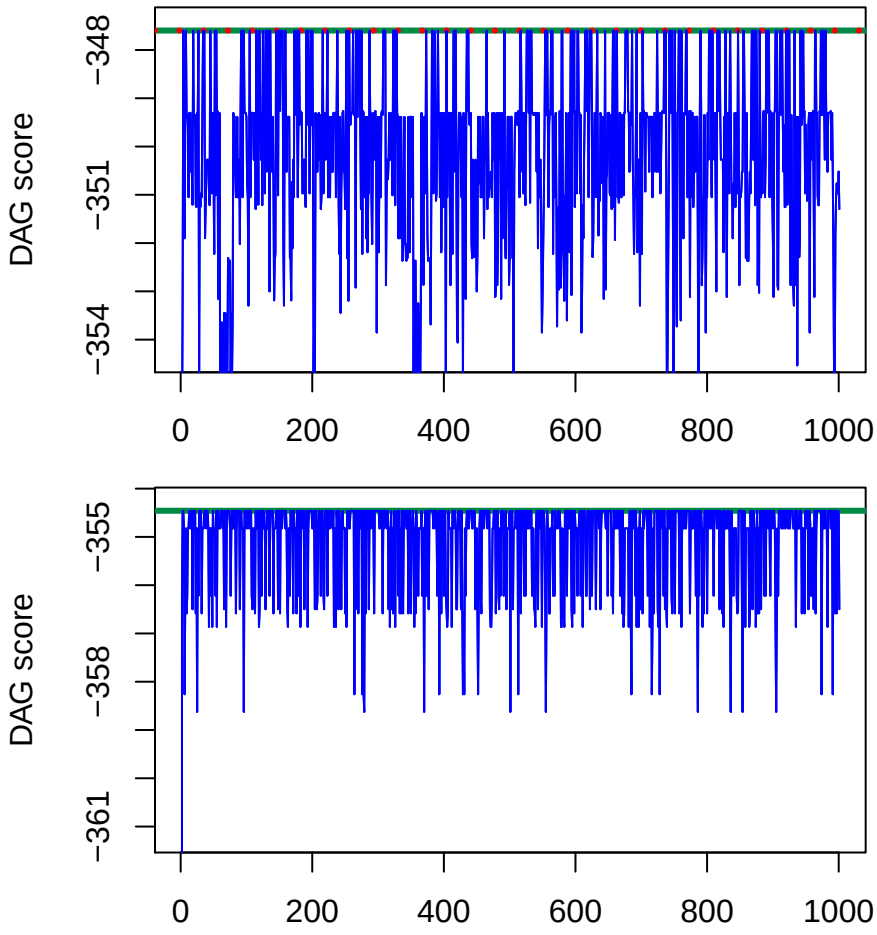
The result of the function is a list containing the sampled DAGs (stored as incidence matrices) as the first element and their scores as the second. We also perform a run with the standard edge reversal moves excluded

```
revallowed <- 0  # disable standard edge reversals
set.seed(seednumber)
example2 <- structureMCMC(n, startDAG, iterations, stepsave, maxparents,
    parenttable, scoretable, revallowed, moveprobs)
```

We can extract the scores and plot them as the runs progress

```
par(mfrow = c(2, 1))  # plotting parameters
par(mar = c(2.5, 5.75, 0.5, 0.75))

nparts <- length(example[[2]])
```

```
maxDAGscore <- max(unlist(example[[2]]))

plot(1:nparts, example[[2]], type = "l", ylab = "DAG score", xlab = "",
    main = "", col = "blue", ylim = c(maxDAGscore - 6.8, maxDAGscore +
        0.2))
abline(h = maxDAGscore, col = "springgreen4", lwd = 3)
abline(h = realDAGlogscore, col = "red", lty = 3, lwd = 3)
lines(1:nparts, example[[2]], type = "l", col = "blue")

maxDAGscore2 <- max(unlist(example2[[2]]))

plot(1:nparts, example2[[2]], type = "l", ylab = "DAG score", xlab = "",
    main = "", col = "blue", ylim = c(maxDAGscore2 - 6.8, maxDAGscore2 +
        0.2))
abline(h = maxDAGscore2, col = "springgreen4", lwd = 3)
abline(h = realDAGlogscore, col = "red", lty = 3, lwd = 3)
lines(1:nparts, example2[[2]], type = "l", col = "blue")
```



2 New edge reversal move

We can include the new edge reversal move, by simply making the vector `moveprobs` have two elements, with the second corresponding to the probability of picking the new edge reversal move.

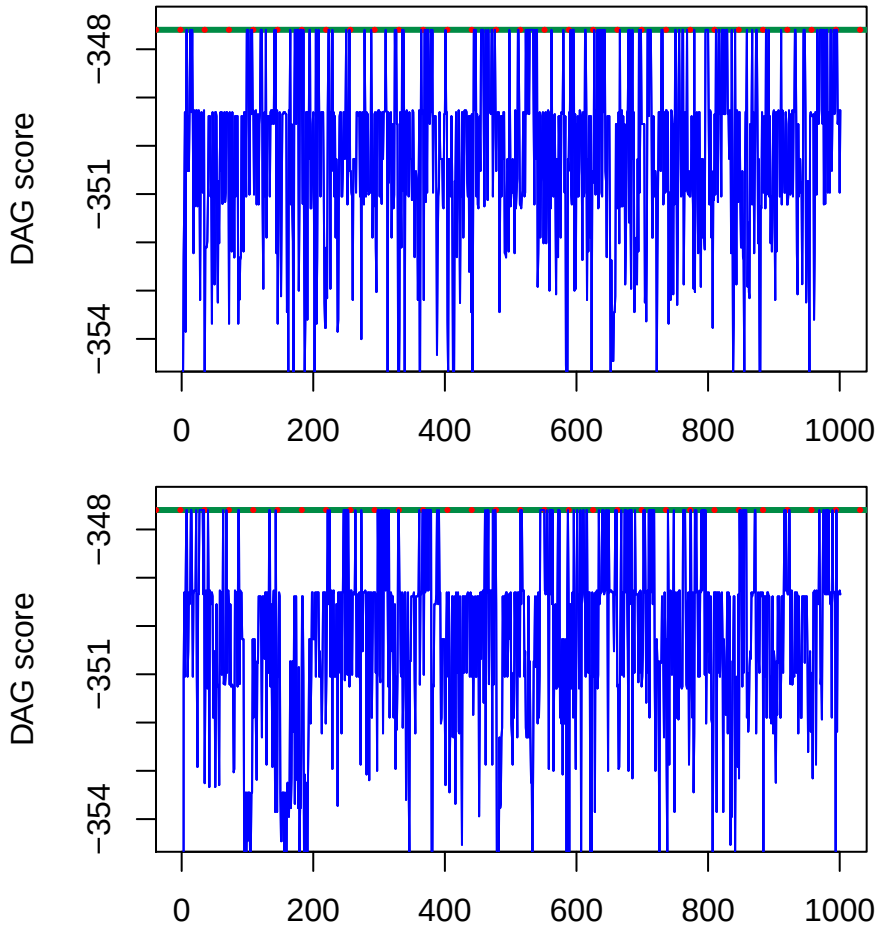
```
moveprobs <- c(0.93, 0.07)  # having length 1 disallows the new edge reversal move
```

Now we run the chains for 40000 iterations, once with the standard edge reversal and once without

```
iterations <- 40000  #number of iterations in the chain
stepsave <- iterations/1000  #how often to save the result
revallowed <- 1  # allow standard edge reversals
set.seed(seednumber)  # set the seed
example <- structureMCMC(n, startDAG, iterations, stepsave, maxparents,
    parenttable, scoretable, revallowed, moveprobs)  # run the MCMC code
revallowed <- 0  # don't allow standard edge reversals
set.seed(seednumber)
example2 <- structureMCMC(n, startDAG, iterations, stepsave, maxparents,
    parenttable, scoretable, revallowed, moveprobs)
```

The trace plot is now

```
par(mfrow = c(2, 1))  # plotting parameters
par(mar = c(2.5, 5.75, 0.5, 0.75))

nparts <- length(example[[2]])
maxDAGscore <- max(unlist(example[[2]]))

plot(1:nparts, example[[2]], type = "l", ylab = "DAG score", xlab = "",
    main = "", col = "blue", ylim = c(maxDAGscore - 6.8, maxDAGscore +
        0.2))
abline(h = maxDAGscore, col = "springgreen4", lwd = 3)
abline(h = realDAGlogscore, col = "red", lty = 3, lwd = 3)
lines(1:nparts, example[[2]], type = "l", col = "blue")

maxDAGscore2 <- max(unlist(example2[[2]]))

plot(1:nparts, example2[[2]], type = "l", ylab = "DAG score", xlab = "",
    main = "", col = "blue", ylim = c(maxDAGscore2 - 6.8, maxDAGscore2 +
        0.2))
abline(h = maxDAGscore2, col = "springgreen4", lwd = 3)
abline(h = realDAGlogscore, col = "red", lty = 3, lwd = 3)
lines(1:nparts, example2[[2]], type = "l", col = "blue")
```



3 Order MCMC

For Order MCMC we now have three elements in the `moveprobs` vector. The first is the probability of swapping any two elements in the order, the second the probability of only swapping adjacent nodes and the last the probability to stay still (to enforce aperiodicity of the chain)

```
prob1 <- 99
if (n > 3) {
    prob1 <- round(6 * 99 * n/(n^2 + 10 * n - 24))
}
prob1 <- prob1/100
moveprobs <- c(prob1, 0.99 - prob1, 0.01)
moveprobs <- moveprobs/sum(moveprobs)  # normalisation
moveprobs

[1] 0.58 0.41 0.01
```

We also need to choose a starting order, for example the identity permutation

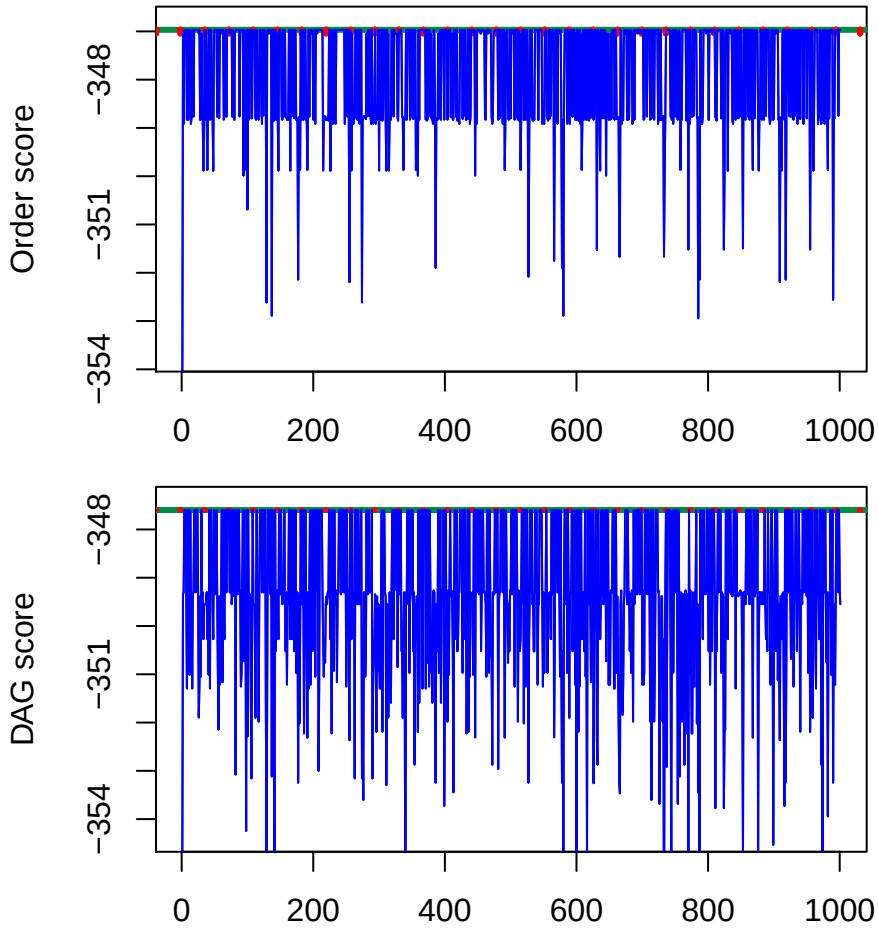
```
startorder <- c(1:n)  # starting order
```

and then we can run the MCMC code for 20000 iterations

```
iterations <- 20000  #number of iterations in the chain
stepsave <- iterations/1000  #how often to save the result
set.seed(seednumber)  # set the seed
example <- orderMCMC(n, startorder, iterations, stepsave, parenttable,
    scoretable, moveprobs)
```

The output now contains four components, a sampled DAG and its score along with the score of the entire order and the sampled order itself (stored as a permutation vector). We can plot the outcome for both the orders and for the DAGs

```
par(mfrow = c(2, 1))  # plotting parameters
par(mar = c(2.5, 5.75, 0.5, 0.75))

nparts <- length(example[[2]])
maxorderscore <- max(unlist(example[[3]]))
maxDAGscore <- max(unlist(example[[2]]))

plot(1:nparts, example[[3]], type = "l", ylab = "Order score",
    xlab = "", main = "", col = "blue", ylim = c(maxorderscore -
        6.8, maxorderscore + 0.2))
abline(h = maxorderscore, col = "springgreen4", lwd = 3)
abline(h = realorderlogscores, col = "red", lty = 3, lwd = 3)
lines(1:nparts, example[[3]], type = "l", col = "blue")

plot(1:nparts, example[[2]], type = "l", ylab = "DAG score", xlab = "",
    main = "", col = "blue", ylim = c(maxDAGscore - 6.8, maxDAGscore +
        0.2))
abline(h = maxDAGscore, col = "springgreen4", lwd = 3)
abline(h = realDAGlogscore, col = "red", lty = 3, lwd = 3)
lines(1:nparts, example[[2]], type = "l", col = "blue")
```



4 Partition MCMC

For Partiton MCMC we have five elements in the `moveprobs` vector. The first is the probability of swapping two nodes from different partititon elements, while the second for swapping nodes from adjacent partition elements. The third corresponds to joining and splitting partition elements with the fourth being the probability of moving a node from one element to another. The last is again the probability of not moving

```
prob1start <- 40/100
prob1 <- prob1start * 100
if (n > 3) {
    prob1 <- round(6 * prob1 * n/(n^2 + 10 * n - 24))
}
prob1 <- prob1/100
prob2start <- 99/100 - prob1start
prob2 <- prob2start * 100
if (n > 3) {
    prob2 <- round(6 * prob2 * n/(n^2 + 10 * n - 24))
}
prob2 <- prob2/100
moveprobs <- c(prob1, prob1start - prob1, prob2start - prob2, prob2,
    0.01)
```

```
moveprobs <- moveprobs/sum(moveprobs)  # normalisation
moveprobs

[1] 0.24 0.16 0.24 0.35 0.01
```

For the starting partition we can choose the DAG with no edges

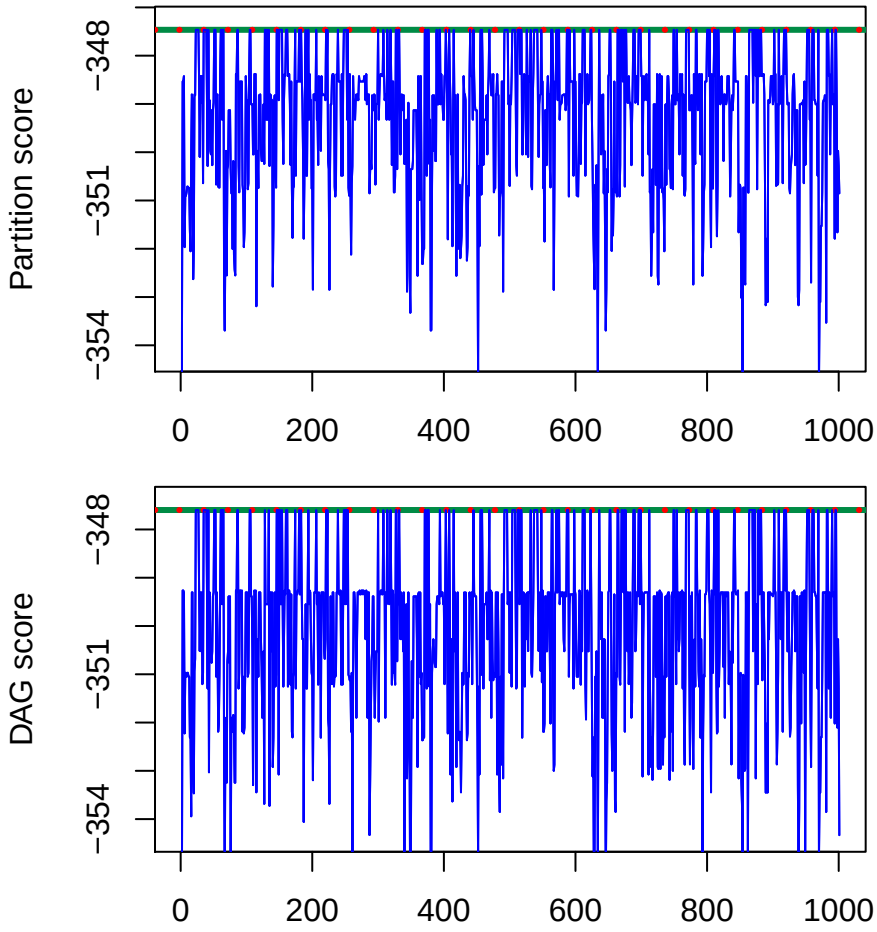
```
startpermutation <- c(1:n)  # pick a starting permutation
startpartition <- c(n)  # and a starting partition - c(n) gives the empty DAG
```

and then run the MCMC code for 10000 iterations

```
iterations <- 10000  #number of iterations in the chain
stepsave <- iterations/1000  #how often to save the result
set.seed(seednumber)  # set the seed
example <- partitionMCMC(n, startpermutation, startpartition, iterations,
    stepsave, parenttable, scoretable, moveprobs)
```

The output now contains a five components: a sampled DAG and its score, the score of the entire partition and the sampled permutation and partition. The trace plot is

```
par(mfrow = c(2, 1))  # plotting parameters
par(mar = c(2.5, 5.75, 0.5, 0.75))

nparts <- length(example[[2]])
maxpartitionscore <- max(unlist(example[[3]]))
maxDAGscore <- max(unlist(example[[2]]))

plot(1:nparts, example[[3]], type = "l", ylab = "Partition score",
    xlab = "", main = "", col = "blue", ylim = c(maxpartitionscore -
        6.8, maxpartitionscore + 0.2))
abline(h = maxpartitionscore, col = "springgreen4", lwd = 3)
abline(h = realpartitionlogscore, col = "red", lty = 3, lwd = 3)
lines(1:nparts, example[[3]], type = "l", col = "blue")

plot(1:nparts, example[[2]], type = "l", ylab = "DAG score", xlab = "",
    main = "", col = "blue", ylim = c(maxDAGscore - 6.8, maxDAGscore +
        0.2))
abline(h = maxDAGscore, col = "springgreen4", lwd = 3)
abline(h = realDAGlogscore, col = "red", lty = 3, lwd = 3)
lines(1:nparts, example[[2]], type = "l", col = "blue")
```



5 Partition MCMC with edge reversal

Finally we can include the new edge reversal move on an underlying Partition MCMC chain by including a sixth element in the `moveprobs` vector, corresponding to the probability of picking the edge reversal move

```
prob1start <- 37/100
prob1 <- prob1start * 100
if (n > 3) {
    prob1 <- round(6 * prob1 * n/(n^2 + 10 * n - 24))
}
prob1 <- prob1/100
prob2start <- 92/100 - prob1start
prob2 <- prob2start * 100
if (n > 3) {
    prob2 <- round(6 * prob2 * n/(n^2 + 10 * n - 24))
}
prob2 <- prob2/100
moveprobs <- c(prob1, prob1start - prob1, prob2start - prob2, prob2,
    0.01, 0.07)
moveprobs <- moveprobs/sum(moveprobs)  # normalisation
moveprobs
```

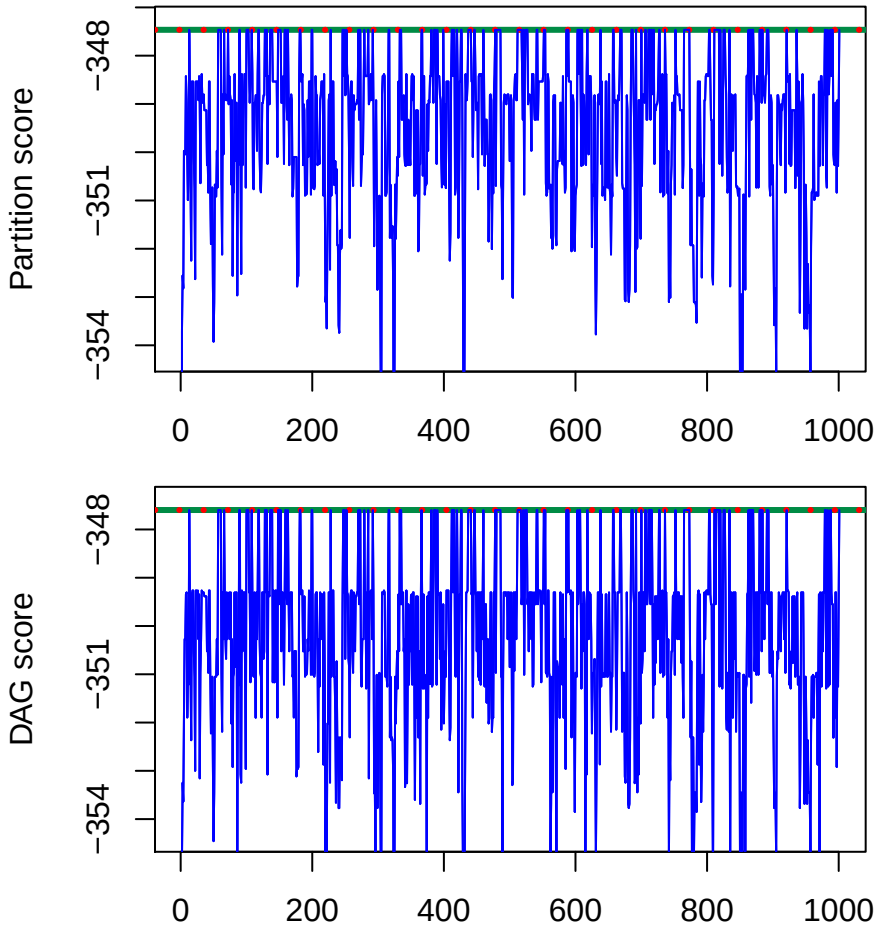
```
[1] 0.22 0.15 0.23 0.32 0.01 0.07
```

This we run for 9000 iterations

```
iterations <- 9000  #number of iterations in the chain
stepsave <- iterations/1000  #how often to save the result
set.seed(seednumber)  # set the seed
example <- partitionMCMC(n, startpermutation, startpartition, iterations,
    stepsave, parenttable, scoretable, moveprobs)
```

and plot the results

```
par(mfrow = c(2, 1))  # plotting parameters
par(mar = c(2.5, 5.75, 0.5, 0.75))

nparts <- length(example[[2]])
maxpartitionscore <- max(unlist(example[[3]]))
maxDAGscore <- max(unlist(example[[2]]))

plot(1:nparts, example[[3]], type = "l", ylab = "Partition score",
    xlab = "", main = "", col = "blue", ylim = c(maxpartitionscore -
        6.8, maxpartitionscore + 0.2))
abline(h = maxpartitionscore, col = "springgreen4", lwd = 3)
abline(h = realpartitionlogscore, col = "red", lty = 3, lwd = 3)
lines(1:nparts, example[[3]], type = "l", col = "blue")

plot(1:nparts, example[[2]], type = "l", ylab = "DAG score", xlab = "",
    main = "", col = "blue", ylim = c(maxDAGscore - 6.8, maxDAGscore +
        0.2))
abline(h = maxDAGscore, col = "springgreen4", lwd = 3)
abline(h = realDAGlogscore, col = "red", lty = 3, lwd = 3)
lines(1:nparts, example[[2]], type = "l", col = "blue")
```


Partition score
−348
−351
−354
0
200
400
600
800
1000
DAG score
−348
−351
−354
0
200
400
600
800
1000