MH-sampler for clone trees

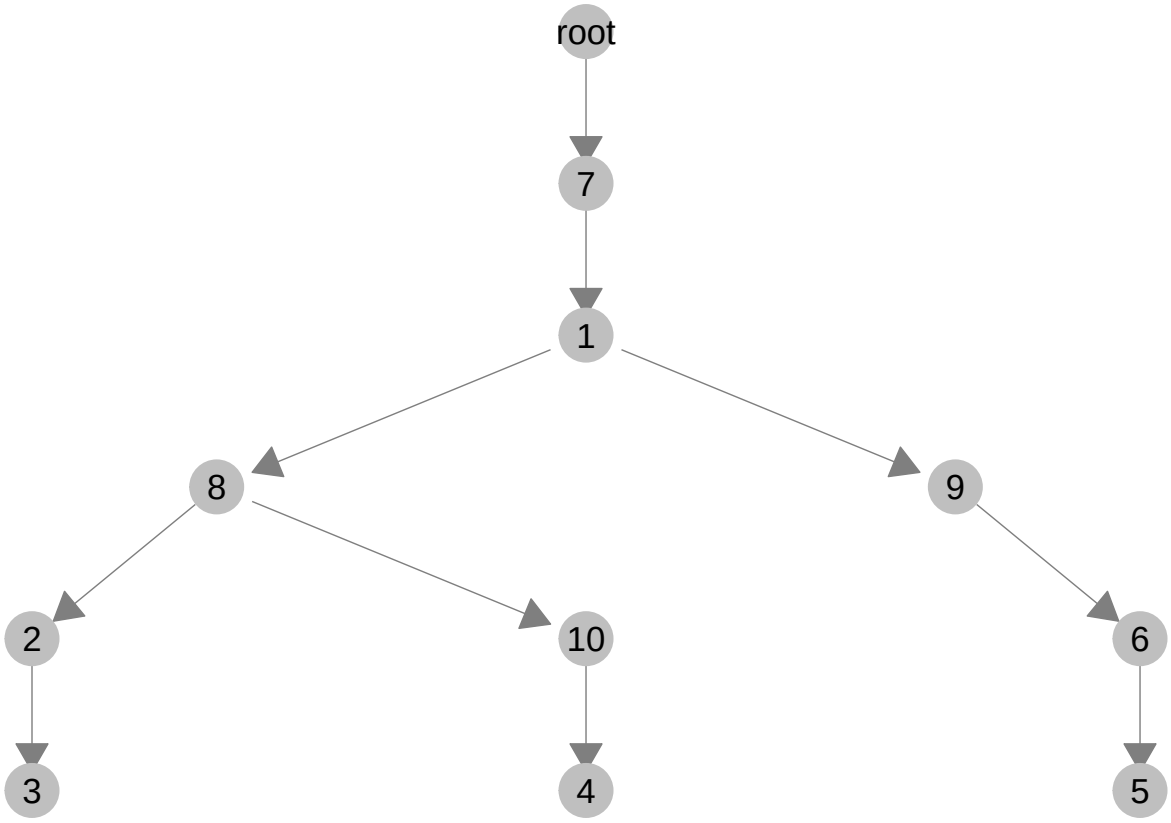
MH-sampler for tree

```
I <- 100; K <- 10; S <- 3
set.seed(123)
test.data <- simulateData(I, K, S)
chains <- readRDS(file.path("..",
                            "output",
                            "cluster_variants.Rmd",
                            "chains.rds"))
w.chain <- chains[grep("w", chains$Parameter), ]
truth <- test.data$w
z.chain <- chains[grep("z", chains$Parameter), ]
mcmc_z <- z.chain %>%
    group_by(Parameter, value) %>%
    summarize(n=n(),
              maxiter=max(Iteration)) %>%
    mutate(probability=n/maxiter)

mcf_stats <- w.chain %>%
    group_by(Parameter) %>%
    summarize(sd=sd(value),
              mean=mean(value))
map_z <- mcmc_z %>%
    group_by(Parameter) %>%
    summarize(value=value[probability==max(probability)])
##mcmc_vals <- orderW(truth, map_z)
##mcmc_w <- matrix(mcmc_vals$mean, 10, 3, byrow=TRUE)
```

```
set.seed(1234)
## construct adjacency matrix
answer <- initializeAdjacencyMatrix(mcf_stats, 0.01)
answer["root", "clone7"] <- answer["clone7", "clone1"] <-
    answer["clone1", "clone8"] <- answer["clone1", "clone9"] <-
    answer["clone8", "clone10"] <- answer["clone8", "clone2"] <-
    answer["clone10", "clone4"] <- answer["clone2", "clone3"] <-
    answer["clone9", "clone6"] <- answer["clone6", "clone5"] <- 1
truth <- plotDAG(answer)
```

```
## Loading required package: scales
```

```
truth
```




```
##
## Each clone has one and only one parent. The parent can be the root.  Practically, this implies that t
##
stopifnot(all(colSums(answer, na.rm=TRUE) == 1))
## cluster precedence order violation matrix
cpov <- create.cpov(mcf_stats)
##trace(calc.tree.fitness, browser)
##calc.tree.fitness(answer, cpov, mcmc_w)
##trace(calc.topology.cost, browser)
TC <- calc.topology.cost(answer, cpov)
MC <- calc.mass.cost(answer, mcfMatrix(mcf_stats))
calc.tree.fitness(answer, cpov, mcfMatrix(mcf_stats))
```

```
## [1] 0.8012409
```

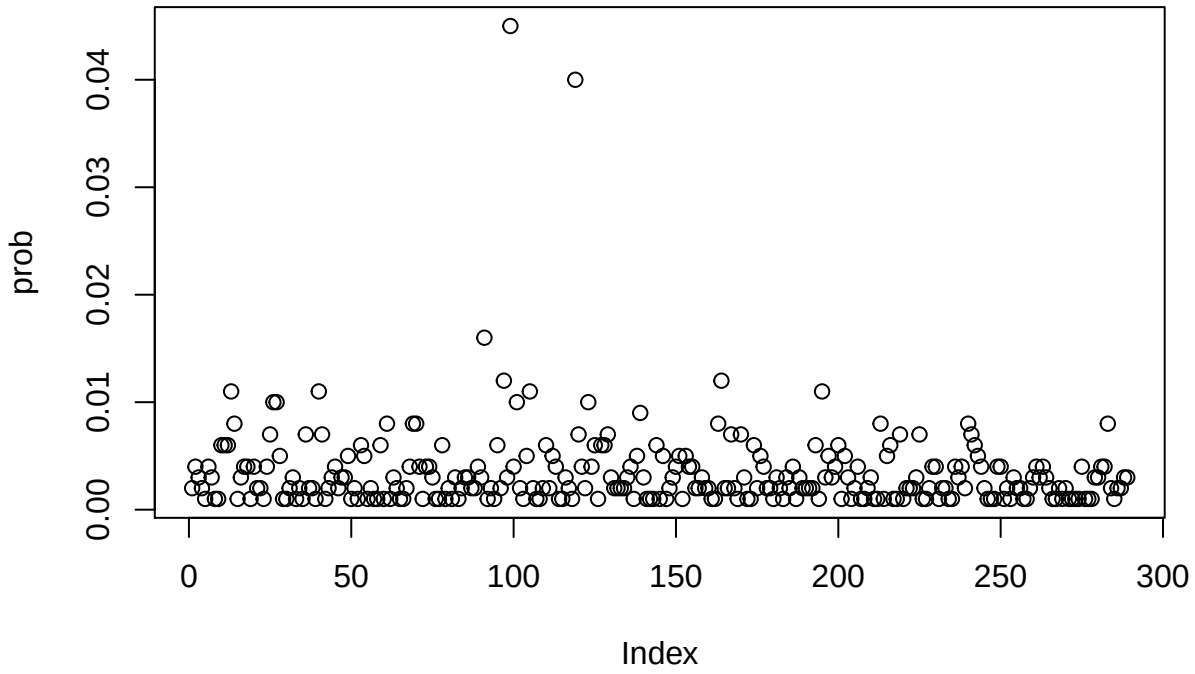
```
##calc.tree.fitness(answer, cpov, w[c(1,2,9,5,10,8,6,3,7,4),])
##admat.chain <- list(init.admat(mcmc_w, zero.thresh=0.01))
## start at best
if(FALSE){
    best.admat.mcmc <- base.admat(mcmc_w)
    best.admat.mcmc[1,1] <- best.admat.mcmc[2,2] <-
        best.admat.mcmc[3,7] <- best.admat.mcmc[3,8] <-
        best.admat.mcmc[5,6] <- best.admat.mcmc[8,9] <-
        best.admat.mcmc[9,4] <- best.admat.mcmc[9,10] <-
        best.admat.mcmc[10,5] <- best.admat.mcmc[11,3] <- 1
}
## shouldn't the truth be the best?'
best.admat.mcmc <- answer
```

```
admat.chain <- list(best.admat.mcmc)
sampled.w.chain <- list(sample.w(w.chain, K))
cpov.chain <- list(create.cpov(mcf_stats = mcf_stats, mcf_matrix = sampled.w.chain[[1]]))
score.chain <- calc.tree.fitness(admat.chain[[1]], cpov.chain[[1]], sampled.w.chain[[1]])

numAccept  <-  0
ncol.to.mutate <- 1
numIter <- 1000
N <- 10
THIN <- 1
#probs <- rep(0, N)
#accept <- rep(0, N)
for (i in seq_len(numIter)) {
#    for(j in seq_len(THIN)){
        fit.prev <- score.chain[i]
        ##
        ## Propose edge
        ##
        admat.star <- mutate.admat(admat.chain[[i]], ncol.to.mutate)
        proposed <- plotDAG(admat.star)
        if(FALSE){
            fig <- arrangeGrob(truth, proposed)
            grid.draw(fig)
        }
        sampled.w.star <- sample.w(w.chain, K)
        cpov.star <- create.cpov(mcf_stats, mcf_matrix = sampled.w.star)
        fit.star <- calc.tree.fitness(admat.star, cpov, sampled.w.star)
        r <- fit.star / fit.prev
        u <- runif(1, 0, 1)
        if(u <= r) {

            admat.chain[[i+1]] <- admat.star
            numAccept <- numAccept + 1
            score.chain <- c(score.chain, fit.star)
            cpov.chain[[i+1]] <- cpov.star
            sampled.w.chain[[i+1]] <- sampled.w.star
        } else {
            admat.chain[[i+1]] <- admat.chain[[i]]
            score.chain <- c(score.chain, fit.prev)
            cpov.chain[[i+1]] <- cpov.chain[[i]]
            sampled.w.chain[[i+1]] <- sampled.w.chain[[i]]
        }
#    }
}
results <- list(admat.chain=admat.chain,
                score.chain=score.chain,
                cpov.chain=cpov.chain,
                sampled.w.chain=sampled.w.chain,
                numAccept=numAccept)
saveRDS(results, file.path("..", "output", "mh_trees_2.Rmd", "trees.rds"))
##
## posterior distribution of trees
##
```

```
numericRepresentation <- function(x){
    x[is.na(x)] <- 0
    x <- as.numeric(x)
    paste(x, collapse="")
}
trees <- sapply(admat.chain, numericRepresentation)
tab <- table(trees)
length(tab)
```

```
## [1] 289
```

```
freq <- as.numeric(tab)
prob <- freq/1000
plot(prob)
```




```
tab2 <- tab[prob > 0.01]
## is the true tree among those with highest probability
tr <- numericRepresentation(answer)
tr %in% names(tab2)
```

```
## [1] FALSE
```

```
acceptRate <- numAccept/(numIter)
acceptRate
```

```
## [1] 0.313
```

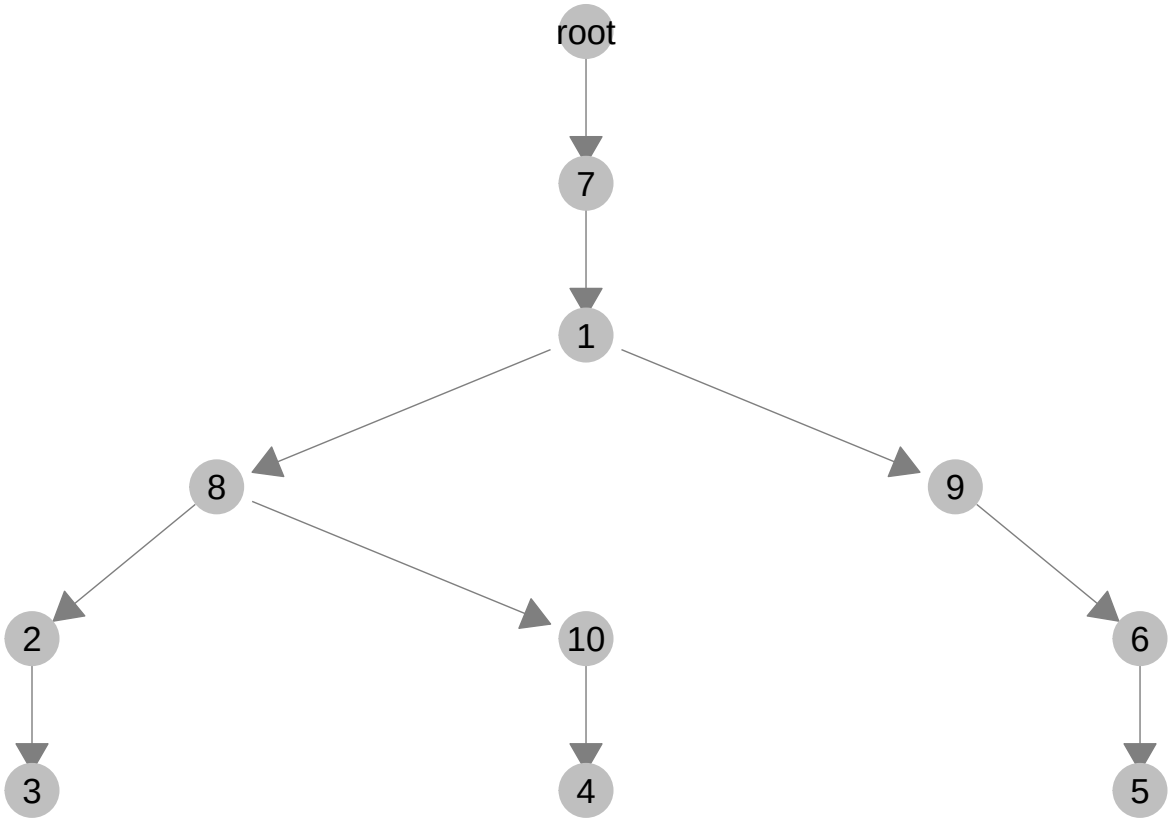
```
max(score.chain)
```

```
## [1] 0.916377
```

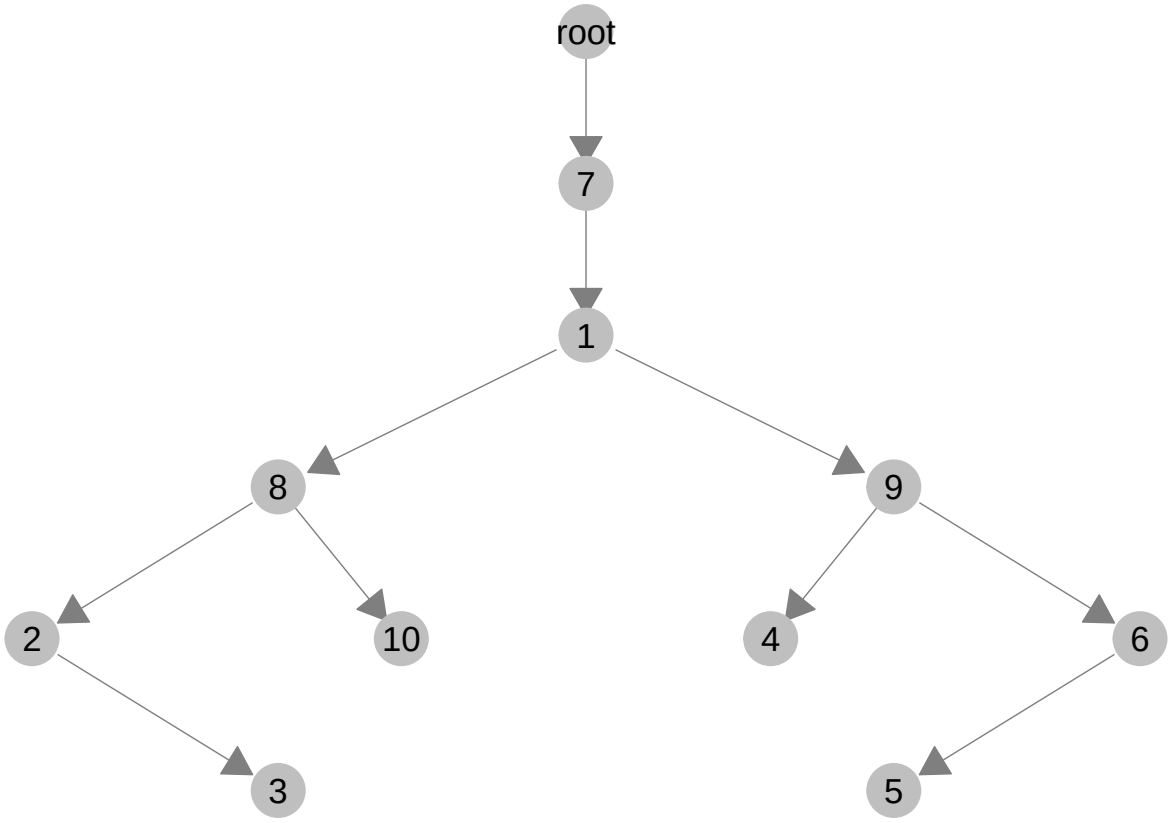
```
max.score.ind <- which(score.chain == max(score.chain))
if(length(max.score.ind) > 1) max.score.ind <- max.score.ind[1]
max.admat <- admat.chain[[max.score.ind]]
max.admat
```

```
##         clone1 clone2 clone3 clone4 clone5 clone6 clone7 clone8 clone9 clone10
## root         0      0      0      0      0      0      1      0      0       0
## clone1      NA      0      0      0      0      0      0      1      1       0
## clone2      NA     NA      1      0     NA     NA     NA      0     NA       0
## clone3      NA     NA     NA     NA     NA     NA     NA     NA     NA      NA
## clone4      NA     NA     NA     NA     NA     NA     NA     NA     NA      NA
## clone5      NA     NA     NA     NA     NA     NA     NA     NA     NA      NA
## clone6      NA     NA     NA      0      1     NA     NA     NA      0      NA
## clone7       1      0      0      0      0      0     NA      0      0       0
## clone8      NA      1      0      0     NA     NA     NA     NA     NA       1
## clone9      NA     NA     NA      0      0      1     NA     NA     NA      NA
## clone10     NA      0      0      1     NA     NA     NA      0     NA      NA
```

```
plotDAG(max.admat)
```




```
mostFreqTree <- names(tab[which(max(prob) == prob)])
mostFreqAdmat <- admat.chain[[which(mostFreqTree == trees)[1]]]
plotDAG(mostFreqAdmat)
```


root
7
1
8
9
2
10
4
6
3
5