Homework10

Jun Rao

10/18/2020

Hidden Markov Models

The goal of this homework is to fit hidden Markov Models on a real data set and compare them to optimal changepoint models.

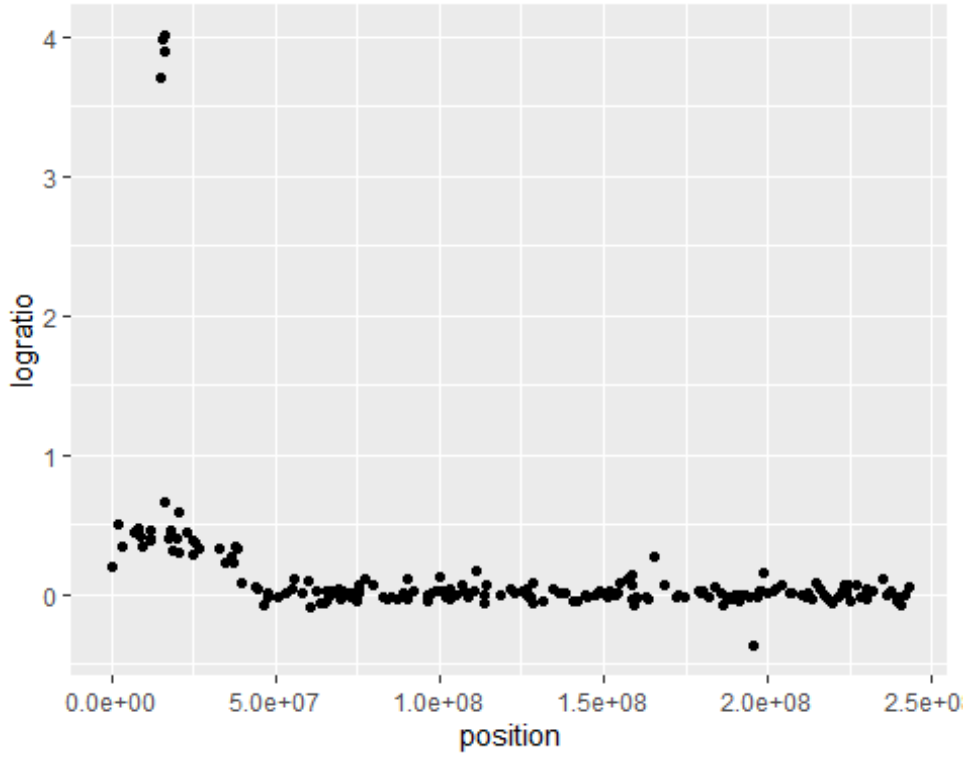
```
library(data.table)
library(ggplot2)
library(stats4)
library(changepoint)
```

1.Plot the profile.id=79 and chromosome=2 from the neuroblastoma data set (x=position, y=logratio). How many observations are there?

```
data(neuroblastoma, package="neuroblastoma")
pro.dt <- data.table(neuroblastoma$profiles)[profile.id=="79" & chromosome=="
2"]
pro.dt
```

```
##      profile.id chromosome  position     logratio
##   1:         79          2     18094  0.200378798
##   2:         79          2   2063049  0.505890930
##   3:         79          2   3098882  0.345964030
##   4:         79          2   7177474  0.449957484
##   5:         79          2   8107742  0.474046599
##  ---
## 192:         79          2 239227603 -0.045431429
## 193:         79          2 239471307 -0.020340448
## 194:         79          2 240618997 -0.081613766
## 195:         79          2 242024751 -0.007231569
## 196:         79          2 242801018  0.048236186
```

```
(gg <- ggplot()+
  geom_point(aes(x=position, y=logratio), data=pro.dt))
```




```
str(pro.dt)
```

```
## Classes 'data.table' and 'data.frame':   196 obs. of  4 variables:
##  $ profile.id: Factor w/ 575 levels "1","2","4","5",..: 76 76 76 76 76 76
76 76 76 76 ...
##  $ chromosome: Factor w/ 24 levels "1","2","3","4",..: 2 2 2 2 2 2 2 2 2 2
 ...
##  $ position  : int  18094 2063049 3098882 7177474 8107742 8179390 8916469
9470069 11607188 11885382 ...
##  $ logratio  : num  0.2 0.506 0.346 0.45 0.474 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

We can find that there are total 196 observations.

2.Use depmixS4 R package to fit a sequence of Gaussian Hidden Markov Models, from nstates=1 to 10. Use the logLik function to compute the negative log likelihood for each model size. Then plot y=negative log likelihood as a function of x=number of states. After what model size is there a kink in the curve? (that is the model size you should select)

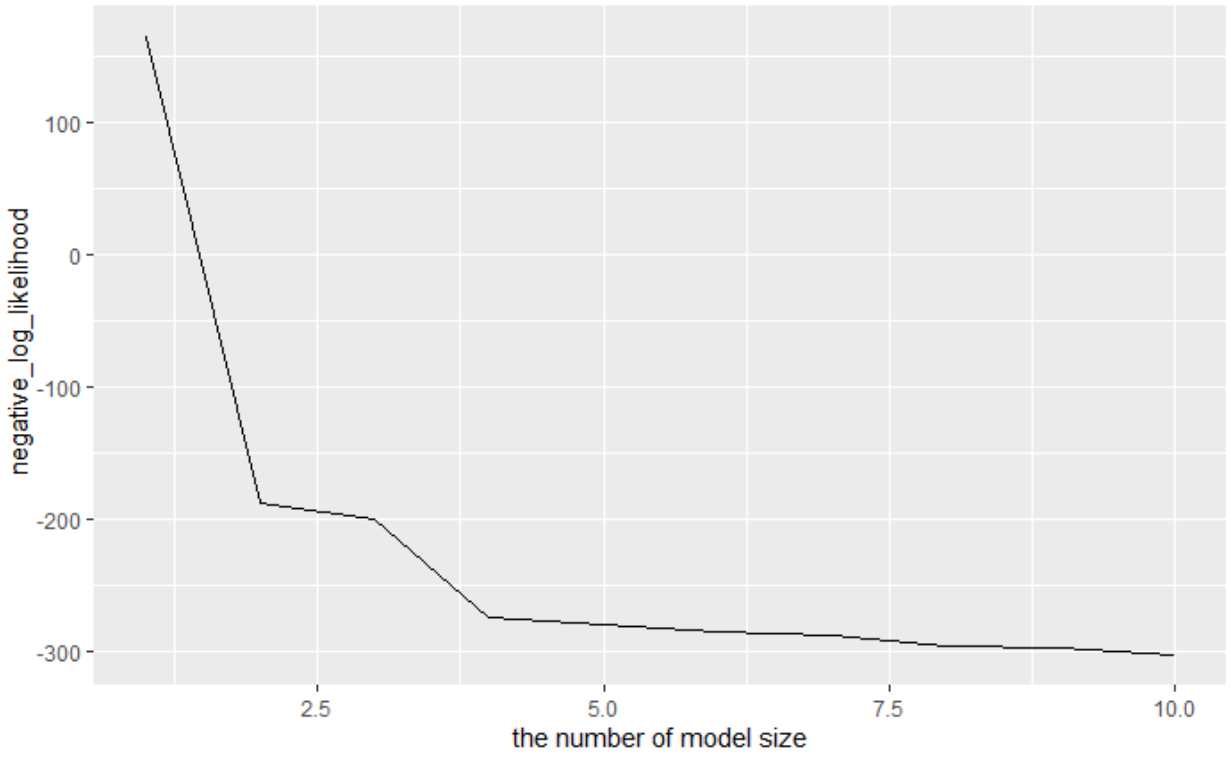
```
total_states = 10
loss.dt.list <- list()
for (nstates in 1:total_states) {
  hmm <- depmixS4::depmix(logratio ~ 1, pro.dt, nstates)
  hmm.fit <- depmixS4::fit(hmm)
  loss.dt.list[[paste(nstates)]] <- data.table(nstates, loss=-depmixS4::logLi
k(hmm.fit))
```

```
}
```

```
## converged at iteration 1 with logLik: -165.3519
## converged at iteration 9 with logLik: 188.4415
## converged at iteration 78 with logLik: 200.6406
## converged at iteration 78 with logLik: 270.1472
## converged at iteration 187 with logLik: 280.6051
```

```
loss.list <- do.call(rbind,  loss.dt.list)
```

```
ggplot()+
  geom_line(aes(nstates,  loss),data=loss.list) +
  labs(x = "the number of model size", y =" negative_log_likelihood")
```



After model size equal 3, there is a kink in the curve

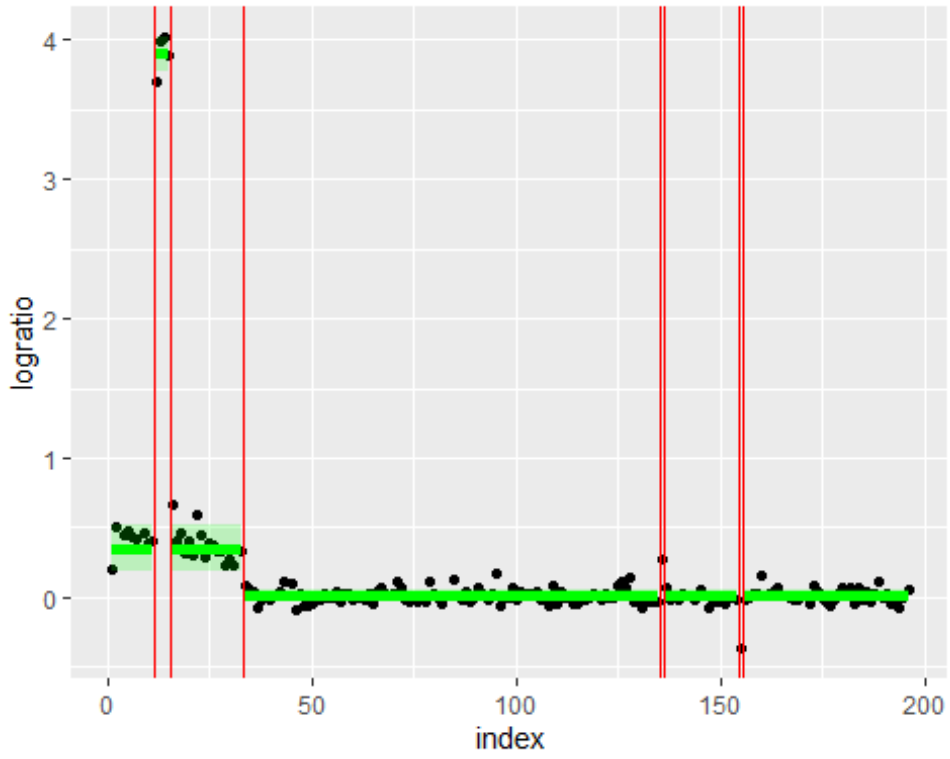
For the selected model size, plot the segmentation on top of the data (for simplicity use x=index instead of position). Make sure to show the mean (geom_segment), the standard deviation (geom_rect), and the changepoints (geom_vline). How many changepoints are there?

```
hmm <- depmixS4::depmix(logratio ~ 1, pro.dt, 3)
hmm.fit <- depmixS4::fit(hmm)
```

```
## converged at iteration 9 with logLik: 251.3187
```

```
resp.dt.list <- list()
for(resp.i in seq_along(hmm.fit@response)){
  param.list <- hmm.fit@response[[resp.i]][[1]]@parameters
  resp.dt.list[[paste(resp.i)]] <- data.table(
    state=resp.i,
    mean=param.list[["coefficients"]],
    sd=param.list[["sd"]])
}


resp.dt <- do.call(rbind, resp.dt.list)
post.dt <- data.table(hmm.fit@posterior)


## rle = run length encoding, for more compressed version of hidden
## markov model, in terms of segments, like optimal changepoint
## models.
state.rle <- rle(post.dt$state)

segs.dt <- data.table(
  n.data=state.rle[["lengths"]],
  state=state.rle[["values"]])
segs.dt[, end := cumsum(n.data)]
segs.dt[, start := c(1, end[-.N]+1)]
segs.with.params <- resp.dt[segs.dt, on="state"]

pro.dt[, index := 1:.N]

ggplot()+
  geom_point(aes(
    index, logratio),
    data=pro.dt)+
  geom_vline(aes(#changes
    xintercept=end+0.5
  ),
  color="red",
  data=segs.with.params[-.N])+
  geom_rect(aes(#sd
    xmin=start, xmax=end,
    ymin=mean-sd, ymax=mean+sd
  ),
  data=segs.with.params,
  fill="green",
  alpha=0.2,
  )+
  geom_segment(aes(#mean
    x=start, xend=end,
    y=mean, yend=mean),
    color="green",
```

```
    size=2,
    data=segs.with.params)
```

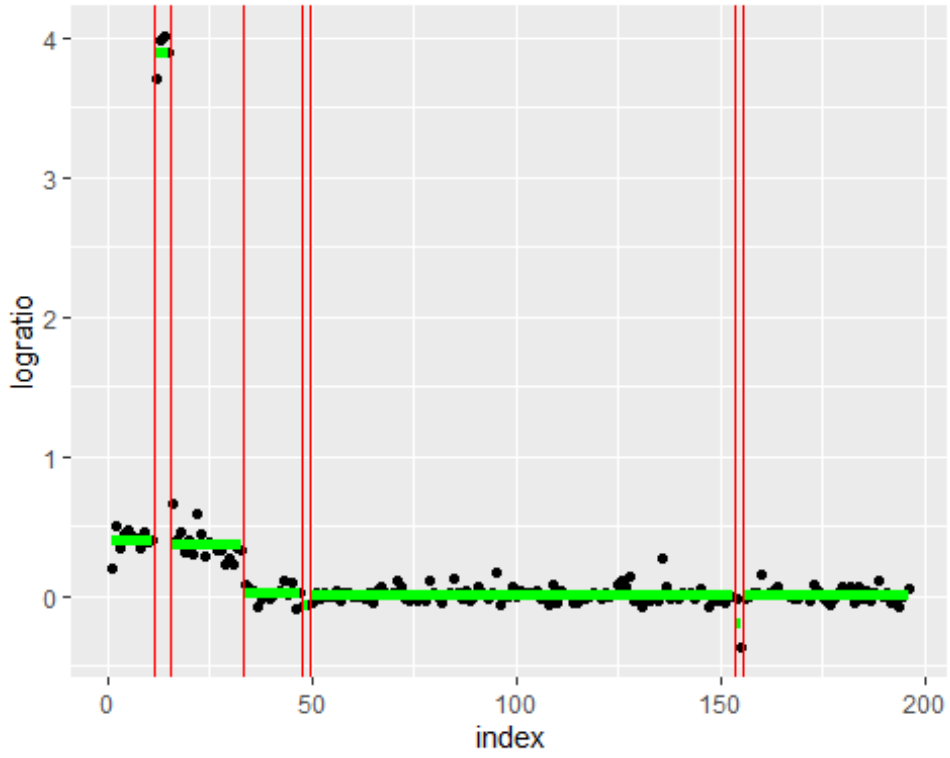


We total have 8 segments, and 7 changepoints

4.Run changepoint::cpt.meanvar(method="SegNeigh", penalty="Manual", Q=the same number of segments as in the HMM) to compute the corresponding optimal changepoint model, and plot that model on top of the data (same as in problem 3). Which model seems to be a better fit, or are they about the same?

```
n.segs <- 8
segs.dt.list <- list()
optimal.models<-changepoint::cpt.meanvar(pro.dt$logratio,penalty="Manual",met
hod="SegNeigh",Q=n.segs)

## Warning in changepoint::cpt.meanvar(pro.dt$logratio, penalty = "Manual", :
## SegNeigh is computationally slow, use PELT instead

## Warning in segneigh.meanvar.norm(c(0.200378797984026, 0.505890929729957, :
 The
## number of segments identified is Q, it is advised to increase Q to make su
re
## changepoints have not been missed.

end <- cpts(optimal.models)
end <- append(end, length(pro.dt$logratio), after = length(end))
```

```
start <- c(1, end[-length(end)]+1)

segs.dt.list[[paste(n.segs)]] <- data.table(start, end)[, .(
    segments=n.segs,
    mean=mean(pro.dt$logratio[start:end]),
    sd = sd(pro.dt$logratio[start:end])
  ), by=.(start, end)]
segs.dt <- do.call(rbind, segs.dt.list)



ggplot()+
  geom_point(aes(
    x=index, y=logratio),
    data=pro.dt)+
  geom_segment(aes(
    x=start, y=mean,
    xend=end, yend=mean),
    color="green",
    size = 2,
    data=segs.dt)+
  geom_vline(aes(
    xintercept=start-0.5),
    color="red",

    data=segs.dt[start>1])
```



There are almost the same. There is a liitle bit difference.But I thougth HMM algorithm fits it better.

The goal is to systematically compare the negative log likelihood of the two packages (changepoint and depmixS4). changepoint::cpt.meanvar with Q segments should always compute the optimal sequence of Q segment mean/variance parameters (Q-1 changepoints), using dynamic programming. depmixS4/HMM always has nstates <= number of segments, so when there are more segments than states the dynamic programming should have a larger likelihood (better fit to data).

Run HMM for nstates=1 to 10, and compute the number of segments in each model. For each model use changepoint::cpt.meanvar(Q=that number of segments).

Save segment mean/sd/start/end parameters for both algorithms and all model sizes in a single data table.

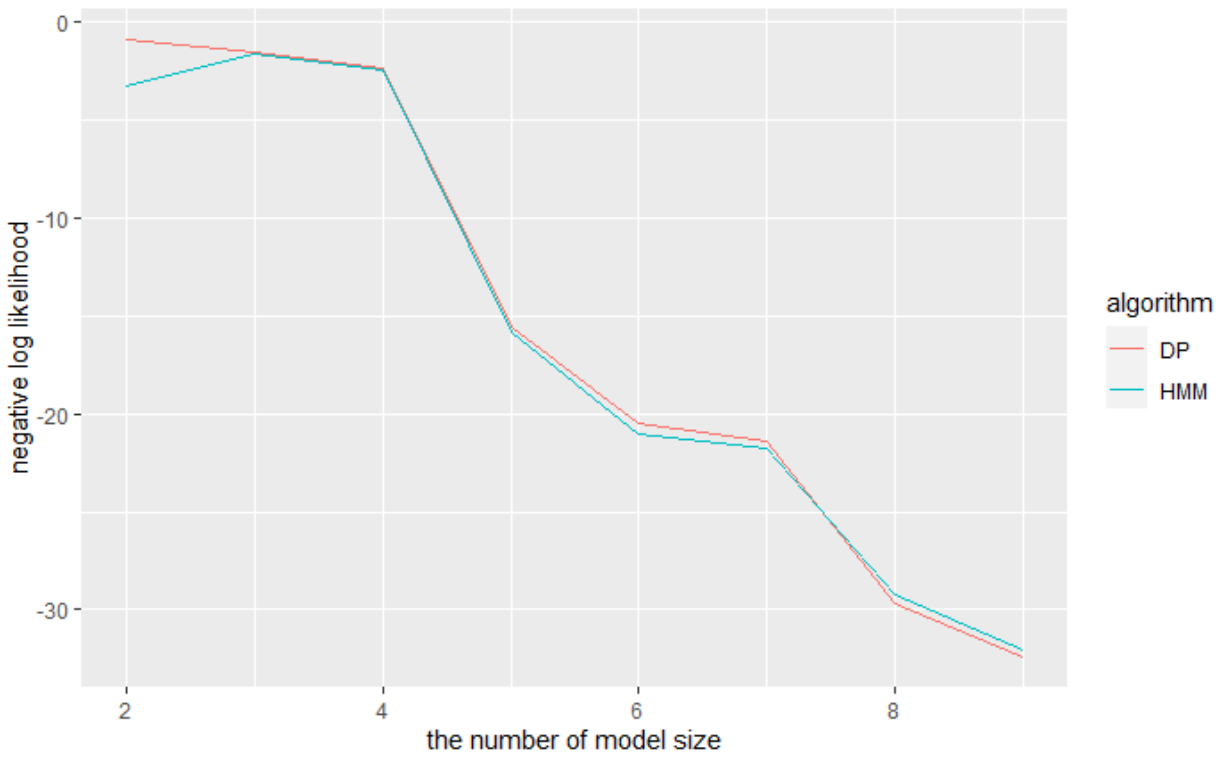
```
result.dt.list <- list()

 total_states <- 9
 for (nstates in 2:total_states){

    opt.segs.dt.list <- list()

    hmm <- depmixS4::depmix(logratio ~ 1, pro.dt, nstates)
    hmm.fit <- depmixS4::fit(hmm)

    resp.dt.list <- list()
    for(resp.i in seq_along(hmm.fit@response)){
      param.list <- hmm.fit@response[[resp.i]][[1]]@parameters
      resp.dt.list[[paste(resp.i)]] <- data.table(
      state=resp.i,
      mean=param.list[["coefficients"]],
      sd=param.list[["sd"]])
    }

    resp.dt <- do.call(rbind, resp.dt.list)
    post.dt <- data.table(hmm.fit@posterior)


    state.rle <- rle(post.dt$state)

    segs.dt <- data.table(
    n.data=state.rle[["lengths"]],
    state=state.rle[["values"]],
    algorithm = 'HMM')
    segs.dt[, end := cumsum(n.data)]
    segs.dt[, start := c(1, end[-.N]+1)]
    segs.with.params <- resp.dt[segs.dt, on="state"]
```

```
    pro.dt[, index := 1:.N]

   hmm.segs.info=segs.with.params[-.N]

   Nsegments <- length(hmm.segs.info$start)



   for (seg.index in 1:Nsegments) {
       start <- hmm.segs.info$start[seg.index]
       end <- hmm.segs.info$end[seg.index]
       mean <- hmm.segs.info$mean[seg.index]
       sd <- hmm.segs.info$sd[seg.index]
       algorithm <- hmm.segs.info$algorithm[seg.index]
      result.dt.list[[paste(nstates,seg.index,start,end,mean,sd,algorithm)]]
<- data.table(nstates,seg.index,start,end,mean,sd,algorithm)
   }

    optimal.models<-changepoint::cpt.meanvar(pro.dt$logratio,penalty="Manual"
,method="SegNeigh",Q=Nsegments)
    opt.end <- cpts(optimal.models)
    opt.end <- append(opt.end, length(pro.dt$logratio), after = length(opt.en
d))

    opt.start <- c(1, opt.end[-length(opt.end)]+1)

    opt.segs.dt.list[[paste(Nsegments)]] <- data.table(opt.start, opt.end)[,
.(
        segments=Nsegments,
        mean=mean(pro.dt$logratio[opt.start:opt.end]),
        sd = sd(pro.dt$logratio[opt.start:opt.end]),
        algorithm = 'OPT'
    ), by=.(opt.start, opt.end)]
  opt.segs.info <- do.call(rbind, opt.segs.dt.list)

     for (seg.index in 1:Nsegments) {
       start <- opt.segs.info$opt.start[seg.index]
       end <- opt.segs.info$opt.end[seg.index]
       mean <- opt.segs.info$mean[seg.index]
       sd <- opt.segs.info$sd[seg.index]
       algorithm <- opt.segs.info$algorithm[seg.index]
      result.dt.list[[paste(nstates,seg.index,start,end,mean,sd,algorithm)]]
<- data.table(nstates,seg.index,start,end,mean,sd,algorithm)
   }
}

head(result.dt <- do.call(rbind, result.dt.list))
```

```
##    nstates seg.index start end        mean         sd algorithm
## 1:       2         1     1  33 0.755056018 1.13626391       HMM
## 2:       2         2    34 135 0.009681706 0.04711407       HMM
## 3:       2         3   136 136 0.755056018 1.13626391       HMM
## 4:       2         4   137 154 0.009681706 0.04711407       HMM
## 5:       2         5   155 155 0.755056018 1.13626391       HMM
## 6:       2         1     1  11 0.401781188 0.08316934       OPT
```

Compute the negative log likelihood using stats::dnorm function, and plot it as a function of the number of segments, different algorithms in different colors (e.g., red=HMM, black=DP). Is the curve lower for DP as expected?

```
hmm.result.dt <- subset(result.dt,result.dt$algorithm == 'HMM')
opt.result.dt <- subset(result.dt,result.dt$algorithm == 'OPT')

neg.log.likelihood.list <- list()

 for (nstates_index in 2:total_states){

  hmm.mean.dt <- subset(hmm.result.dt,hmm.result.dt$nstates==nstates_index)$m
ean
  neg_log_L = -sum(dnorm(hmm.mean.dt, mean = mean(hmm.mean.dt), sd = sd(hmm.m
ean.dt)))
  algorithm <- 'HMM'
  neg.log.likelihood.list[[paste(nstates_index,neg_log_L,algorithm)]] <- data
.table(nstates_index,neg_log_L,algorithm)

  opt.mean.dt <- subset(opt.result.dt,opt.result.dt$nstates==nstates_index)$m
ean
  neg_log_L = -sum(dnorm(opt.mean.dt, mean = mean(opt.mean.dt), sd = sd(opt.m
ean.dt)))
  algorithm <- 'DP'
  neg.log.likelihood.list[[paste(nstates_index,neg_log_L,algorithm)]] <- data
.table(nstates_index,neg_log_L,algorithm)

 }


 neg.log.likelihood.dt <- do.call(rbind, neg.log.likelihood.list)


ggplot()+
  geom_line(aes(
    nstates_index, neg_log_L, color=algorithm),
    data= neg.log.likelihood.dt)+
  xlab("the number of model size") + ylab("negative log likelihood")
```



These two algorithms are almost same. DP performance as we expected