lab2_hmm_WKAD

Wuhao Wang,Kangbo Chen,Hugo Axandersson,Daniel Persson

2022-09-13

Contribution

We discuss every question and we just check and use combined code.

Question 1

```
rm(list=ls())
states = as.character(1:10)

symbol = as.character(1:10)

start_prob = rep(0.1,10)

# equal probability to step forward or stay
trans_prob = matrix(
  c(0.5,0.5,0,0,0,0,0,0,0,0,
    0,0.5,0.5,0,0,0,0,0,0,0,
    0,0,0.5,0.5,0,0,0,0,0,0,
    0,0,0,0.5,0.5,0,0,0,0,0,
    0,0,0,0,0.5,0.5,0,0,0,0,
    0,0,0,0,0,0.5,0.5,0,0,0,
    0,0,0,0,0,0,0.5,0.5,0,0,
    0,0,0,0,0,0,0,0.5,0.5,0,
    0,0,0,0,0,0,0,0,0.5,0.5,
    0.5,0,0,0,0,0,0,0,0,0.5),nrow = 10,byrow = TRUE
)

# equal probability to report from [i-2,i+2]
emission_prob = matrix(
  c(0.2,0.2,0.2,0,0,0,0,0,0.2,0.2,
    0.2,0.2,0.2,0.2,0,0,0,0,0,0.2,
    0.2,0.2,0.2,0.2,0.2,0,0,0,0,0,
    0,0.2,0.2,0.2,0.2,0.2,0,0,0,0,
    0,0,0.2,0.2,0.2,0.2,0.2,0,0,0,
    0,0,0,0.2,0.2,0.2,0.2,0.2,0,0,
    0,0,0,0,0.2,0.2,0.2,0.2,0.2,0,
    0,0,0,0,0,0.2,0.2,0.2,0.2,0.2,
    0.2,0,0,0,0,0,0.2,0.2,0.2,0.2,
    0.2,0.2,0,0,0,0,0,0.2,0.2,0.2),nrow = 10,byrow = TRUE
)
```

```
hmm = initHMM(states, symbol,start_prob, trans_prob,emission_prob)
```

Question 2

The code below will generate 100 time steps.

```
set.seed(123456789)
sample = simHMM(hmm,100)
sample
```

```
## $states
##   [1] "8"  "9"  "9"  "9"  "9"  "9"  "10" "10" "1"  "2"  "3"  "4"  "5"  "6"  "6"
##  [16] "6"  "6"  "6"  "7"  "8"  "8"  "9"  "9"  "10" "1"  "1"  "2"  "3"  "4"  "5"
##  [31] "6"  "7"  "7"  "8"  "9"  "10" "10" "1"  "2"  "3"  "4"  "5"  "5"  "5"  "6"
##  [46] "7"  "8"  "8"  "8"  "8"  "9"  "9"  "9"  "10" "1"  "2"  "3"  "3"  "3"  "4"
##  [61] "5"  "5"  "6"  "6"  "6"  "6"  "7"  "8"  "9"  "9"  "9"  "9"  "10" "10" "10"
##  [76] "1"  "2"  "2"  "2"  "2"  "2"  "2"  "3"  "3"  "4"  "5"  "5"  "5"  "5"  "5"
##  [91] "6"  "7"  "7"  "8"  "8"  "8"  "9"  "10" "1"  "2"
##
## $observation
##   [1] "10" "10" "1"  "1"  "8"  "9"  "8"  "8"  "10" "4"  "2"  "3"  "3"  "5"  "6"
##  [16] "7"  "8"  "7"  "6"  "7"  "7"  "7"  "10" "9"  "1"  "3"  "4"  "2"  "3"  "4"
##  [31] "4"  "5"  "6"  "8"  "8"  "8"  "9"  "1"  "4"  "3"  "3"  "6"  "4"  "7"  "6"
##  [46] "7"  "9"  "8"  "7"  "7"  "7"  "7"  "7"  "2"  "10" "10" "4"  "1"  "5"  "4"
##  [61] "6"  "7"  "4"  "4"  "6"  "8"  "5"  "6"  "8"  "9"  "8"  "7"  "10" "10" "2"
##  [76] "9"  "10" "1"  "1"  "1"  "10" "2"  "5"  "2"  "3"  "7"  "6"  "7"  "3"  "5"
##  [91] "4"  "7"  "9"  "7"  "6"  "7"  "9"  "10" "10" "1"
```

Question 3

Below is the code to compute `filtered,smoothed probability distribution` and `path` .

```
obs = sample$observation

# use this function to compute proportion in a column
f <- function(x)
{

  return(round(x/sum(x),2))
}

# the result from forward is log form, we need to use exp
alpha = exp(forward(hmm, obs))
filter  = apply(alpha,2,f)

beta = exp(backward(hmm,obs))
smooth = alpha*beta
smooth = apply(smooth,2,f)

# another way to do this use posterior function
```

```
#
# smooth = posterior(hmm,obs)
# filter = prop.table(exp(forward(hmm,obs)),2)

path = viterbi(hmm, obs)

cat('The most probable path')
```

```
## The most probable path
```

```
cat('\n')
```

```
path
```

```
##   [1] "8"  "8"  "9"  "9"  "9"  "9"  "9"  "10" "1"  "2"  "2"  "2"  "2"  "3"  "4"
##  [16] "5"  "6"  "6"  "6"  "7"  "8"  "9"  "10" "1"  "1"  "1"  "2"  "2"  "3"  "4"
##  [31] "5"  "6"  "7"  "8"  "9"  "10" "1"  "1"  "2"  "2"  "3"  "4"  "4"  "5"  "5"
##  [46] "6"  "7"  "7"  "7"  "7"  "7"  "8"  "9"  "10" "1"  "1"  "2"  "2"  "3"  "3"
##  [61] "4"  "5"  "5"  "5"  "5"  "6"  "6"  "6"  "6"  "7"  "8"  "9"  "10" "1"  "1"
##  [76] "1"  "1"  "1"  "1"  "1"  "1"  "2"  "3"  "3"  "4"  "5"  "5"  "5"  "5"  "5"
##  [91] "5"  "6"  "7"  "7"  "8"  "9"  "10" "1"  "1"  "1"
```

```
cat('Part of smoothed distribution')
```

```
## Part of smoothed distribution
```

```
cat('\n')
```

```
smooth[,1:20]
```

```
##       index
## states    1    2    3    4    5    6    7 8 9 10   11   12   13   14   15   16
##     1  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0 1  0 0.00 0.00 0.00 0.00 0.00 0.00
##     2  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0 0  1 0.37 0.11 0.02 0.00 0.00 0.00
##     3  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0 0  0 0.63 0.53 0.27 0.07 0.00 0.00
##     4  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0 0  0 0.00 0.36 0.53 0.43 0.19 0.00
##     5  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0 0  0 0.00 0.00 0.18 0.42 0.52 0.37
##     6  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0 0  0 0.00 0.00 0.00 0.08 0.27 0.50
##     7  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0 0  0 0.00 0.00 0.00 0.00 0.02 0.12
##     8  0.58 0.26 0.00 0.00 0.00 0.00 0.00 0 0  0 0.00 0.00 0.00 0.00 0.00 0.00
##     9  0.37 0.63 0.79 0.63 0.47 0.32 0.16 0 0  0 0.00 0.00 0.00 0.00 0.00 0.00
##     10 0.05 0.11 0.21 0.37 0.53 0.68 0.84 1 0  0 0.00 0.00 0.00 0.00 0.00 0.00
##       index
## states   17   18   19   20
##     1  0.00 0.00 0.00 0.00
##     2  0.00 0.00 0.00 0.00
##     3  0.00 0.00 0.00 0.00
##     4  0.00 0.00 0.00 0.00
##     5  0.00 0.00 0.00 0.00
##     6  0.65 0.30 0.10 0.02
```

```
##     7  0.31 0.57 0.51 0.29
##     8  0.03 0.13 0.39 0.57
##     9  0.00 0.00 0.00 0.12
##     10 0.00 0.00 0.00 0.00
```

```
cat('Part of filtered distribution')
```

```
## Part of filtered distribution
```

```
cat('\n')
```

```
filter[,1:20]
```

```
##       index
## states   1    2    3    4    5    6    7    8    9 10  11   12   13   14   15
##     1  0.2 0.22 0.24 0.25 0.00 0.38 0.00 0.00 0.43  0 0.0 0.00 0.00 0.00 0.00
##     2  0.2 0.22 0.24 0.25 0.00 0.00 0.00 0.00 0.00  1 0.5 0.25 0.13 0.00 0.00
##     3  0.0 0.00 0.12 0.19 0.00 0.00 0.00 0.00 0.00  0 0.5 0.50 0.37 0.27 0.00
##     4  0.0 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  0 0.0 0.25 0.37 0.40 0.38
##     5  0.0 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  0 0.0 0.00 0.13 0.27 0.38
##     6  0.0 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  0 0.0 0.00 0.00 0.07 0.19
##     7  0.0 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  0 0.0 0.00 0.00 0.00 0.04
##     8  0.2 0.11 0.00 0.00 0.00 0.00 0.00 0.00 0.00  0 0.0 0.00 0.00 0.00 0.00
##     9  0.2 0.22 0.18 0.09 0.23 0.12 0.16 0.14 0.07  0 0.0 0.00 0.00 0.00 0.00
##     10 0.2 0.22 0.24 0.22 0.77 0.50 0.84 0.86 0.50  0 0.0 0.00 0.00 0.00 0.00
##       index
## states   16   17   18   19   20
##     1  0.00 0.00 0.00 0.00 0.00
##     2  0.00 0.00 0.00 0.00 0.00
##     3  0.00 0.00 0.00 0.00 0.00
##     4  0.00 0.00 0.00 0.00 0.00
##     5  0.48 0.00 0.00 0.00 0.00
##     6  0.36 0.55 0.28 0.17 0.08
##     7  0.14 0.33 0.44 0.43 0.30
##     8  0.02 0.11 0.22 0.40 0.42
##     9  0.00 0.02 0.06 0.00 0.20
##     10 0.00 0.00 0.00 0.00 0.00
```

Question 4

```
standard = sample$states
filter_pre = apply(filter,2, function(x) names(which.max(x)))
smooth_pre = apply(smooth,2, function(x) names(which.max(x)))

get_acc <-function(x1,x2)
{
  t <- table(x1, x2)
  acc <- sum(diag(t))/ sum(t)
  return(acc)
}

print(paste0("The accuracy of filter: ",get_acc(standard, filter_pre)))
```

```
## [1] "The accuracy of filter: 0.52"
```

```
cat("\n")
```

```
print(paste0("The accuracy of smooth: ",get_acc(standard, smooth_pre)))
```

```
## [1] "The accuracy of smooth: 0.66"
```

```
cat("\n")
```

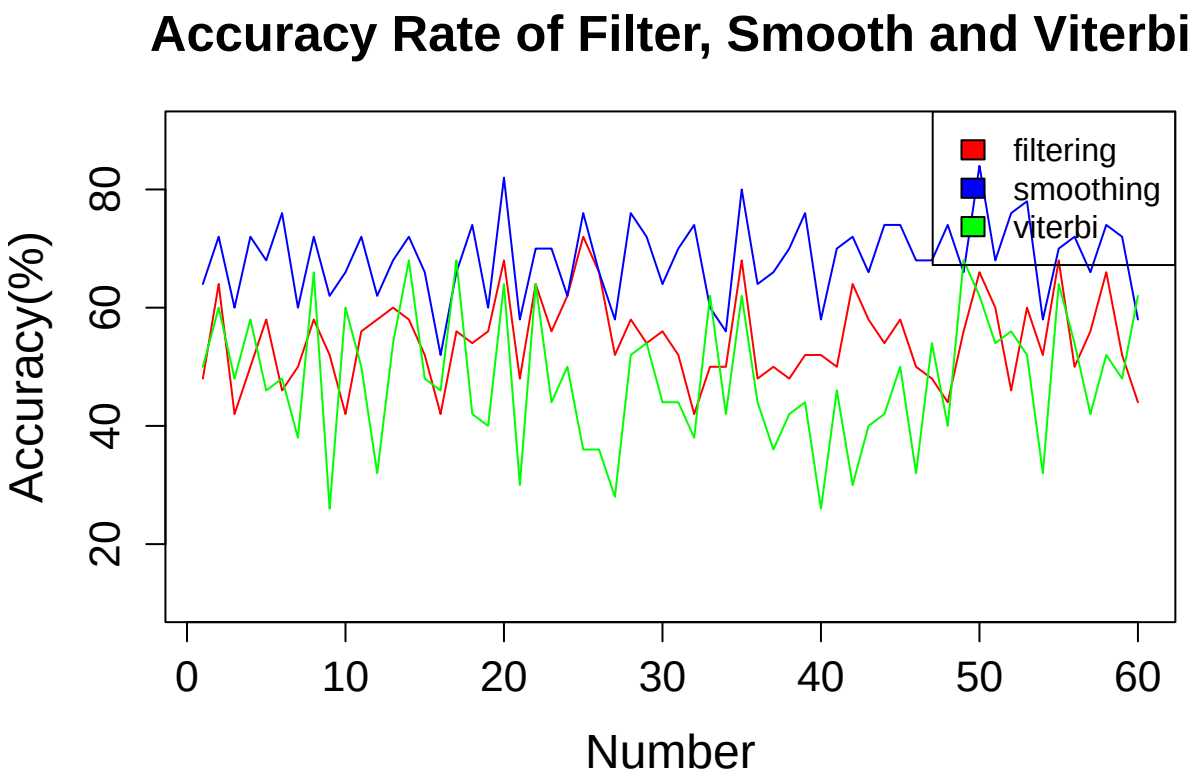
```
print(paste0("The accuracy of path: ",get_acc(standard, path)))
```

```
## [1] "The accuracy of path: 0.43"
```

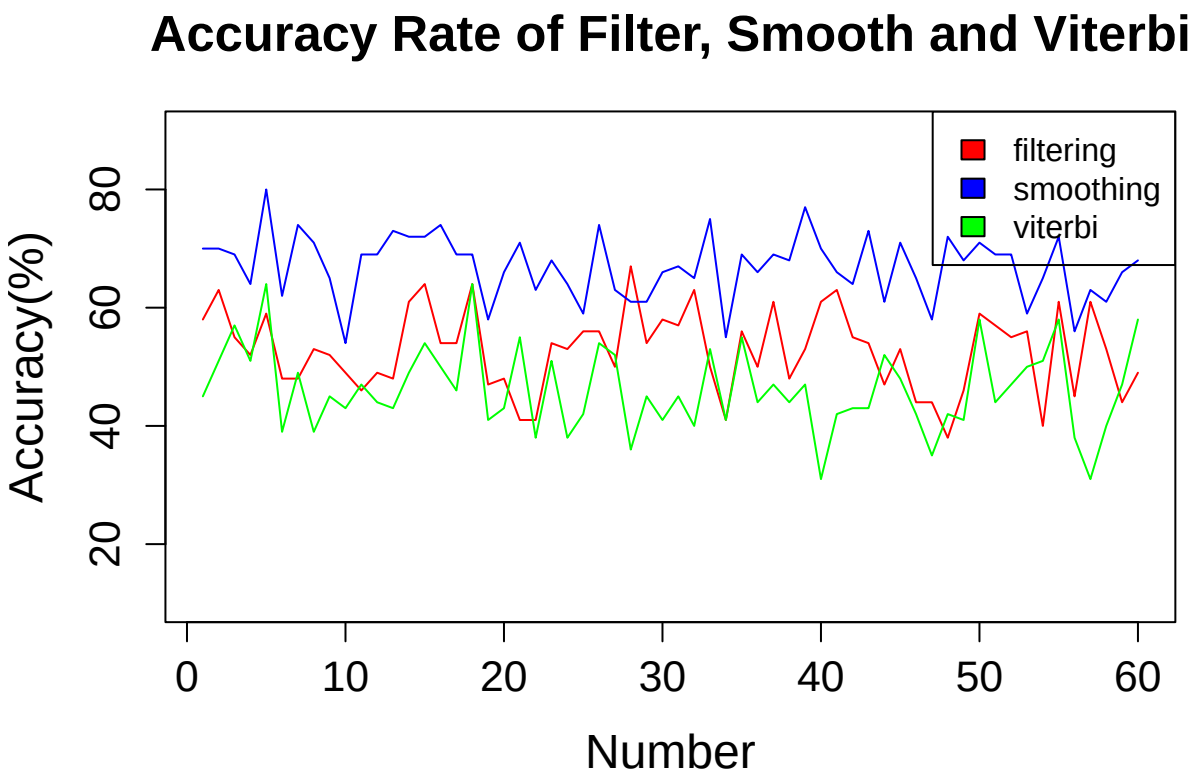
Question 5

```
set.seed(123456789)

Sim_HMM <- function(model, len){

  # simulation
  sim_HMM_model <- simHMM(model,length = len)
  observation <- sim_HMM_model$observation

  # Filtered distribution
  filter_HMM <- forward(model, observation)
  filter_HMM <- proportions(exp(filter_HMM),2)

  # Smoothed distribution
  smooth_HMM <- posterior(model, observation)

  # Most Probable Configuration
  Vertbi_path <- viterbi(model, observation)

  # state prediction
  filter_state <- apply(X = filter_HMM, MARGIN = 2, FUN = function(x){names(which.max(x))})
  Smooth_state <- apply(X = smooth_HMM, MARGIN = 2, FUN = function(x){names(which.max(x))})

  # accuracy
  acc_filter <- sum(sim_HMM_model$states == filter_state)/len*100
  acc_smooth <- sum(sim_HMM_model$states == Smooth_state)/len*100
  acc_vertbi <- sum(sim_HMM_model$states == Vertbi_path)/len*100

  return(list('filter_sta' = acc_filter, 'smooth_sta' = acc_smooth, 'vertbi_sta' = acc_vertbi ))
}

sampling_sim_HMM <- function(model,len, samp_num){

  res_filt <- c()
  res_smooth <- c()
```

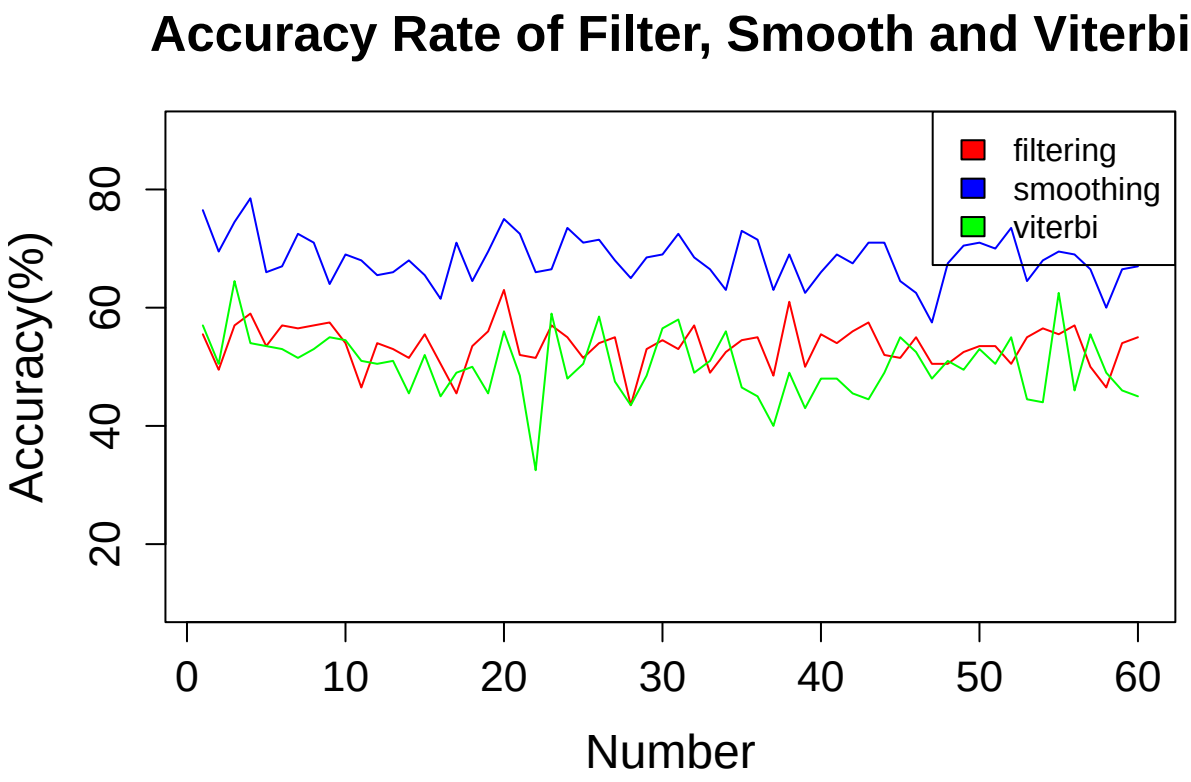
```
  res_viter <- c()

  for (i in 1:samp_num) {

    res <- Sim_HMM(model, len)
    res_filt[i] <- res[[1]]
    res_smooth[i] <- res[[2]]
    res_viter[i] <- res[[3]]

  }

  return(list('filter'=res_filt, 'smooth'=res_smooth,'viter'=res_viter))
}

# res <- sampling_sim_HMM(HMM_model,100,60)

plot_func <- function(results){

  plot(results$filter,type = 'l',col = 'red',ylim = c(10,90),
       xlab = 'Number',ylab = 'Accuracy(%)',
       main = 'Accuracy Rate of Filter, Smooth and Viterbi',
       cex.main = 1.6, cex.axis=1.3, cex.lab = 1.5)
  lines(results$smooth,type = 'l',col = 'blue')
  lines(results$viter,type = 'l',col = 'green')
  legend('topright',
         legend = c('filtering','smoothing','viterbi'),
         fill = c('red','blue','green'))

}

res_50 <- sampling_sim_HMM(hmm,50,60)
res_100 <- sampling_sim_HMM(hmm,100,60)
res_200 <- sampling_sim_HMM(hmm,200,60)

plot_func(res_50)
```




```
plot_func(res_100)
```




```
plot_func(res_200)
```



$$Filter \ = \ p(z^t|x^{0:t})$$

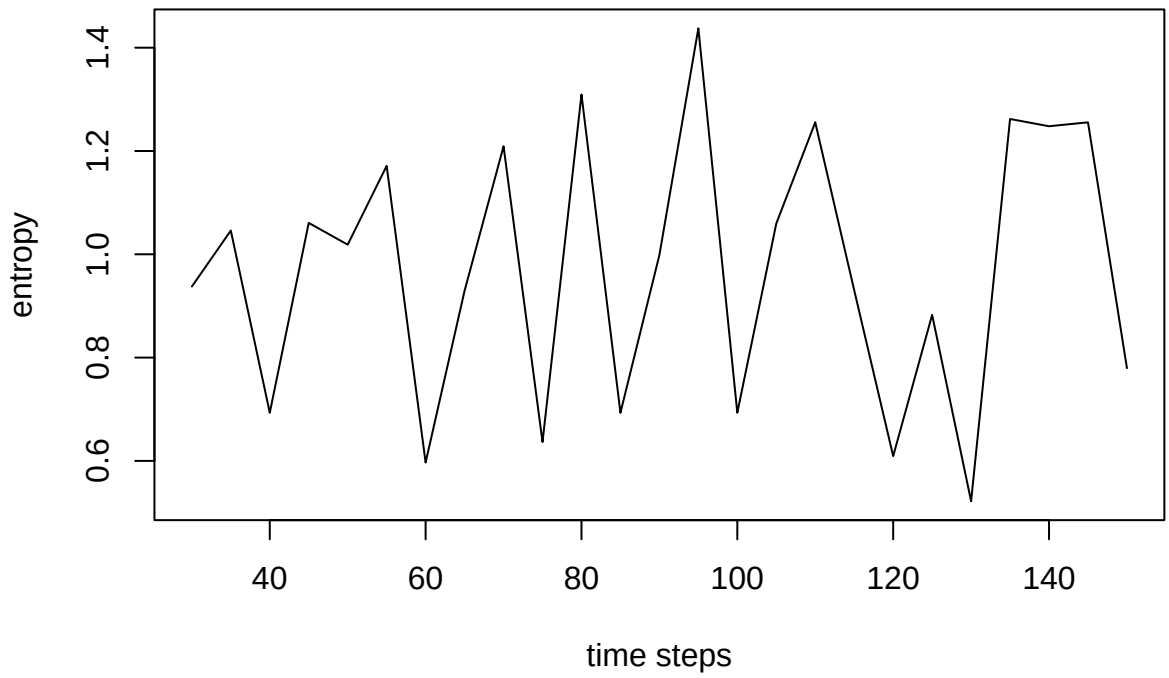
$$Smooth \ = \ p(z^t|x^{0:T})$$

Compare two equation above, we can see that smooth process can refer to more observation when do prediction on same thing, which means smoothing can have more information than filtering. That's the reason why it's more precise.

The `Viterbi Algorithm` is maximizing joint distribution of all Z (from $z^0 to z^T$), which means sometimes it will not chose the hidden state with highest probability but the most proper one regarding the whole path . So it will show less accuracy than smooth.

Question 6

```
set.seed(123456789)
sample <- simHMM(hmm, 150)$observation
forward <- lapply(seq(30, 150, 5), function(x) forward(hmm, sample[1:x]))
filter <- lapply(forward, function(x) prop.table(exp(x), 2))
res <- mapply(function(x, y) entropy.empirical(x[, y]), filter, seq(30, 150, 5))
plot(x = seq(30, 150, 5), unlist(res), type = "l",
xlab = "time steps",
ylab = "entropy")
```



Entropy can describe uncertainty of a system and higher entropy means higher uncertainty. From the graph, we can see that the with time steps goes increase (more observations), entropy always fluctuate. Under such circumstance, we can not say we can know better about robot's location when we have more observation.

Question 7

We want to derive the expression below

$$P(Z_{101}|X_{1:100})$$

Notice that, this can be re-write as expression below according to basic statistics.

$$\sum_{Z_{100}} P(Z_{100}, Z_{101}|X_{1:100})$$

And we can use the rules of conditional probability

$$\sum_{Z_{100}} P(Z_{100}, Z_{101}|X_{1:100}) = \sum_{Z_{100}} P(Z_{100}|X_{1:100})P(Z_{101}|Z_{100})$$

Notice that $P(Z_{100}|X_{1:100})$ is our filter at time stamp 100, $P(Z_{101}|Z_{100})$ is the transition probability we used in initializing HMM model. So, $P(Z_{101}|X_{1:100})$ is the linear combination of transition matrix with filtering distribution $P(Z_{100}|X_{1:100})$, which in the row space of transition matrix.

```
## [1] "The probabilites of the hidden states for the time step 101 is:"
```

```
##  [1] 0.31818182 0.21212121 0.07575758 0.01010101 0.00000000 0.00000000
##  [7] 0.00000000 0.00000000 0.10606061 0.27777778
```