Lab2GroupReport

aswma317, akssr921,varsi146

2023-09-21

Question 1

```
####1: Build a HMM####
num_states <- 10#There are 10 possible States(unobserved)
num_symbols <- 10#There are 5 possible Symbols, but 10 states(observed)

#Uniform initial probability for each state
initial_probs <- rep(1/num_states, num_states)

#Transition probabilities(Unobserved states to unobserved states)
transition_probs <- matrix(0, num_states, num_states)
for (i in 1:num_states) {
  #It has 2 choices: move right or stay. Cannot move backward
  transition_probs[i,i] <- 1/2

  right <- ifelse(i+1 == 11, 1, i+1)#Considers circular reference
  transition_probs[i,right] <- 1/2
}
transition_probs
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]  0.5  0.5  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.0
##  [2,]  0.0  0.5  0.5  0.0  0.0  0.0  0.0  0.0  0.0   0.0
##  [3,]  0.0  0.0  0.5  0.5  0.0  0.0  0.0  0.0  0.0   0.0
##  [4,]  0.0  0.0  0.0  0.5  0.5  0.0  0.0  0.0  0.0   0.0
##  [5,]  0.0  0.0  0.0  0.0  0.5  0.5  0.0  0.0  0.0   0.0
##  [6,]  0.0  0.0  0.0  0.0  0.0  0.5  0.5  0.0  0.0   0.0
##  [7,]  0.0  0.0  0.0  0.0  0.0  0.0  0.5  0.5  0.0   0.0
##  [8,]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.5  0.5   0.0
##  [9,]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.5   0.5
## [10,]  0.5  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.5
```

```
#Emission probabilities(Unobserved states to observed symbols)
emission_probs <- 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), num_states, num_states)
emission_probs
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]  0.2  0.2  0.2  0.0  0.0  0.0  0.0  0.0  0.2   0.2
##  [2,]  0.2  0.2  0.2  0.2  0.0  0.0  0.0  0.0  0.0   0.2
##  [3,]  0.2  0.2  0.2  0.2  0.2  0.0  0.0  0.0  0.0   0.0
##  [4,]  0.0  0.2  0.2  0.2  0.2  0.2  0.0  0.0  0.0   0.0
##  [5,]  0.0  0.0  0.2  0.2  0.2  0.2  0.2  0.0  0.0   0.0
##  [6,]  0.0  0.0  0.0  0.2  0.2  0.2  0.2  0.2  0.0   0.0
##  [7,]  0.0  0.0  0.0  0.0  0.2  0.2  0.2  0.2  0.2   0.0
##  [8,]  0.0  0.0  0.0  0.0  0.0  0.2  0.2  0.2  0.2   0.2
##  [9,]  0.2  0.0  0.0  0.0  0.0  0.0  0.2  0.2  0.2   0.2
## [10,]  0.2  0.2  0.0  0.0  0.0  0.0  0.0  0.2  0.2   0.2
```

```
# Create the HMM
hmm_model <- initHMM(States = as.character(1:num_states), Symbols = as.character(1:num_symbols),
                     startProbs = initial_probs,
                     transProbs = transition_probs,
                     emissionProbs = emission_probs)

print(hmm_model)
```

```
## $States
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
##
## $Symbols
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
##
## $startProbs
##   1   2   3   4   5   6   7   8   9  10
## 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##
## $transProbs
##      to
## from   1   2   3   4   5   6   7   8   9  10
##   1  0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
##   2  0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0
##   3  0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0
##   4  0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0
##   5  0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0
##   6  0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0
##   7  0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0
##   8  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0
##   9  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5
##   10 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5
##
## $emissionProbs
##       symbols
## states   1   2   3   4   5   6   7   8   9  10
```

```
##     1  0.2 0.2 0.2 0.0 0.0 0.0 0.0 0.0 0.2 0.2
##     2  0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.0 0.0 0.2
##     3  0.2 0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.0 0.0
##     4  0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.0
##     5  0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0 0.0
##     6  0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0
##     7  0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0
##     8  0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2
##     9  0.2 0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.2
##     10 0.2 0.2 0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.2
```

Question 2

```
####2: Simulate HMM for 100 steps####
# Simulate the HMM for 100 time steps
run_simulation <- function(num_steps, hmm_model){
  #set.seed(123)
  simulated_data <- simHMM(hmm_model, length = num_steps)

# All the observations are within [i - 2, i + 2] distance
  return(simulated_data)
}
num_steps <- 100 #Time steps
simulated_data <- run_simulation(num_steps, hmm_model)

# Extract the sequence of states and observations
states_sequence <- simulated_data$states
observations_sequence <- simulated_data$observation

#Print the sequences (states and observations)
print("States Sequence:")
```

```
## [1] "States Sequence:"
```

```
states_sequence
```

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

```
print("Observations Sequence:")
```

```
## [1] "Observations Sequence:"
```

```
observations_sequence
```

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

Question 3

Below is the code for the filtered and smoothed probability distributions for each of the 100 time point and also to compute the most probable path.

```
sample = simHMM(hmm_model,100)
fil_smooth <- function(sam){
  observed <- sam$observation
    #calculate alpha
  alpha <-  exp(forward(hmm_model, observed))
    #filtering
  fil <-  apply(alpha,2,prop.table)
    #calculating beta
  beta <-  exp(backward(hmm_model,observed))

    #smoothing
  smo <-  alpha*beta
  smooth <- apply(smo,2,prop.table)
  return(list(fil,smooth))
}


#filtering
filter <-  fil_smooth(sample)[[1]]

smooth <- fil_smooth(sample)[[2]]

#path of viterbi algorithm
path = viterbi(hmm_model, sample$observation)


cat('The most Probable Path is')
```

```
## The most Probable Path is
```

```
path
```

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

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

Filtered and smoothed probability distribution values can be seen below

```
filter[,1:10]
```

```
##       index
## states   1         2         3         4          5          6         7
##     1  0.0 0.0000000 0.0000000 0.0000000 0.00000000 0.00000000 0.0000000
##     2  0.0 0.0000000 0.0000000 0.0000000 0.00000000 0.00000000 0.0000000
##     3  0.2 0.1111111 0.0000000 0.0000000 0.00000000 0.00000000 0.0000000
##     4  0.2 0.2222222 0.0000000 0.0000000 0.00000000 0.00000000 0.0000000
##     5  0.2 0.2222222 0.2857143 0.1538462 0.07692308 0.05128205 0.0000000
##     6  0.2 0.2222222 0.2857143 0.3076923 0.23076923 0.20512821 0.0000000
##     7  0.2 0.2222222 0.2857143 0.3076923 0.30769231 0.35897436 0.3333333
##     8  0.0 0.0000000 0.1428571 0.2307692 0.26923077 0.38461538 0.4393939
##     9  0.0 0.0000000 0.0000000 0.0000000 0.11538462 0.00000000 0.2272727
##     10 0.0 0.0000000 0.0000000 0.0000000 0.00000000 0.00000000 0.0000000
##       index
## states         8          9        10
##     1  0.0000000 0.00000000 0.0000000
##     2  0.0000000 0.00000000 0.0000000
##     3  0.0000000 0.00000000 0.0000000
##     4  0.0000000 0.00000000 0.0000000
##     5  0.0000000 0.00000000 0.0000000
##     6  0.0000000 0.00000000 0.0000000
##     7  0.1666667 0.08835341 0.0501139
##     8  0.3863636 0.29317269 0.2164009
##     9  0.3333333 0.38152610 0.3826879
##     10 0.1136364 0.23694779 0.3507973
```

```
smooth[,1:10]
```

```
##       index
## states          1         2          3          4          5         6
##     1  0.00000000 0.0000000 0.00000000 0.00000000 0.00000000 0.0000000
##     2  0.00000000 0.0000000 0.00000000 0.00000000 0.00000000 0.0000000
##     3  0.08356913 0.0000000 0.00000000 0.00000000 0.00000000 0.0000000
##     4  0.26611500 0.1671383 0.00000000 0.00000000 0.00000000 0.0000000
##     5  0.34161662 0.3650918 0.33427650 0.11986432 0.02806585 0.0000000
##     6  0.23388500 0.3181415 0.39590701 0.42882437 0.27539540 0.1122634
##     7  0.07481426 0.1496285 0.24037593 0.36298965 0.49045488 0.4461283
##     8  0.00000000 0.0000000 0.02944055 0.08832166 0.20608387 0.4416083
##     9  0.00000000 0.0000000 0.00000000 0.00000000 0.00000000 0.0000000
##     10 0.00000000 0.0000000 0.00000000 0.00000000 0.00000000 0.0000000
##       index
## states         7          8          9          10
##     1  0.0000000 0.00000000 0.00000000 0.000000000
##     2  0.0000000 0.00000000 0.00000000 0.000000000
##     3  0.0000000 0.00000000 0.00000000 0.000000000
##     4  0.0000000 0.00000000 0.00000000 0.000000000
```

```
##      5  0.0000000 0.00000000 0.00000000 0.000000000
##      6  0.0000000 0.00000000 0.00000000 0.000000000
##      7  0.3087244 0.10579503 0.02631815 0.004341073
##      8  0.5171681 0.47042706 0.26371873 0.094901028
##      9  0.1741076 0.37881019 0.53309011 0.439089353
##      10 0.0000000 0.04496772 0.17687301 0.461668546
```

Question 4

```
#function to return the states name from probability table
fil_smooth_pred <- function(x){
  sta <- rep(NA,ncol(x))
  for(i in 1:ncol(x)){
    sta[i] <- names(which.max(x[,i]))
  }
  return(sta)
}

#getting the predicted states
filter_pred <- fil_smooth_pred(filter)
smooth_pred <- fil_smooth_pred(smooth)


#function to calculate accuracy
cal_accuracy <- function(pre,act){
  t <- table(pre,act)
  acc <- sum(diag(t))/ sum(t)
  return(acc)
}

#accuracy of filtering
acc_filter <- cal_accuracy(filter_pred,sample$states)

#accuracy of smoothing
acc_smooth <- cal_accuracy(smooth_pred,sample$states)

#viterbi accuracy
acc_viterbi <- cal_accuracy(path,sample$states)

cat('Accuracy of filtered probability distributions is',acc_filter*100,'%')
```

```
## Accuracy of filtered probability distributions is 42 %
```

```
cat('Accuracy of smoothed probability distributions is',acc_smooth*100,'%')
```

```
## Accuracy of smoothed probability distributions is 59 %
```

```
cat('Accuracy of most probable path is',acc_viterbi*100,'%')
```

```
## Accuracy of most probable path is 53 %
```

Question 5

```
simulate_HMM <- function(model, time_steps, alt_method = FALSE){
  sim_timeStep <- simHMM(model, length = time_steps)
  sim_actual <- sim_timeStep$states
  sim_obs <- sim_timeStep$observation

  Alpha <- exp(forward(hmm = model, observation = sim_obs))
  Beta <- exp(backward(hmm = model, observation = sim_obs))

  filtered <- apply(Alpha,2,prop.table)
  filter_pred <- apply(filtered, 2, which.max)
  filter_pred <- sapply(filter_pred, function(x) paste0('s',x))

  if (alt_method == FALSE) {
    smoothed <- Alpha*Beta
    smoothed <- apply(smoothed, 2, prop.table)
    smoothed_pred <- apply(smoothed, 2, which.max)
    smoothed_pred <- sapply(smoothed_pred, function(x) paste0('s',x))
  }else{
    # Alternative method to calculate smoothed distribution. Correct?
    smoothed <- posterior(hmm = model, observation = sim_obs)
    smoothed_pred <- apply(smoothed, 2, which.max)
    smoothed_pred <- sapply(smoothed_pred, function(x) paste0('s',x))
  }


  viterbi_path <- viterbi(hmm = model, observation = sim_obs)

  return(list('actual' = sim_actual, 'obs' = sim_obs,'filter_pred'= filter_pred,
              'smoothed_pred' = smoothed_pred,
              'viterbi' = viterbi_path, 'filter_mat' = filtered))
}

accuracy <- function(actual, pred){
  accuracy <- sum(diag(table(actual, pred)))/sum(table(actual, pred))
  return(accuracy)
}

sim100_results <- simulate_HMM(model = hmm_model1, time_steps = 100, alt_method = FALSE)

sim100_samplesRes <- list(
  filter_accs = NULL,
  smooth_accs = NULL,
  viterbi_accs = NULL
)

for (i in 1:100) {
  simulate_hmm_vals <- simulate_HMM(model = hmm_model1, time_steps = 100, alt_method = FALSE)
  sim100_samplesRes$filter_accs[i] <- accuracy(actual = simulate_hmm_vals$actual,
                                               pred = simulate_hmm_vals$filter_pred)

  sim100_samplesRes$smooth_accs[i] <- accuracy(actual = simulate_hmm_vals$actual,
```

```
                                            pred = simulate_hmm_vals$smoothed_pred)

  sim100_samplesRes$viterbi_accs[i] <- accuracy(actual = simulate_hmm_vals$actual,
                                            pred = simulate_hmm_vals$viterbi)
}


cat('The mean accuracy of filtered distribution is', mean(sim100_samplesRes$filter_accs), '\n')
```

```
## The mean accuracy of filtered distribution is 0.539
```

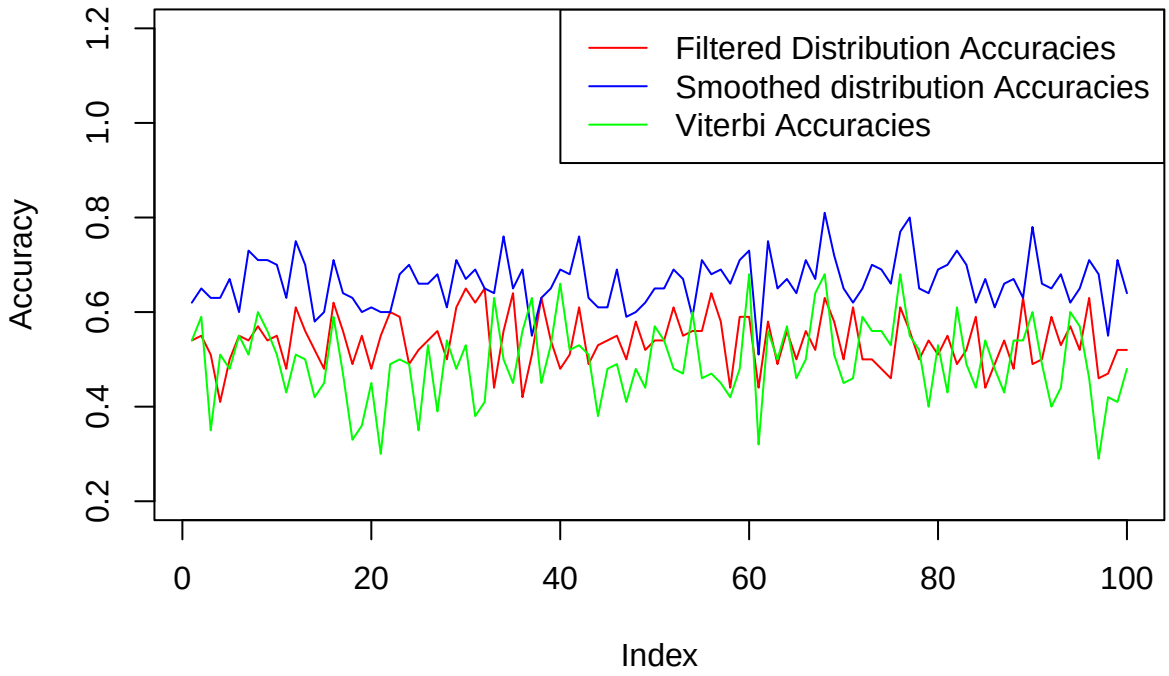
```
cat('The mean accuracy of smoothed distribution is', mean(sim100_samplesRes$smooth_accs), '\n')
```

```
## The mean accuracy of smoothed distribution is 0.6646
```

```
cat('The mean accuracy of most probable paths distribution is', mean(sim100_samplesRes$viterbi_accs), '
```

```
## The mean accuracy of most probable paths distribution is 0.496
```

```
plot(sim100_samplesRes$filter_accs, type = 'l',
     col = 'red', ylim = c(0.2, 1.2), ylab = 'Accuracy')
lines(sim100_samplesRes$smooth_accs, type = 'l', col = 'blue')
lines(sim100_samplesRes$viterbi_accs, type = 'l', col = 'green')
# Adding Legend
legend("topright", legend = c("Filtered Distribution Accuracies", "Smoothed distribution Accuracies", "
       col = c("red", "blue", "green"), lty = 1)
```



From the results above, we can see that in general, smoothed distributions are more accurate compared to filtered distributions, and this can be represented mathematically as well.

Smoothing is given by the following expression: $p(z^t|x^{0:T})$

While, filtering is given by the following expression: $p(z^t|x^{0:t})$

T in the smoothing expression means that the entire process has finished running and we calculate the probability of the hidden states given all the observations from start to the end of the process. While, the expression for filtering has t implying that the probability of hidden states are calculated only using the observations upto time step t and is hence more sensitive to short-term fluctuations or noisy data. In general, smoothed distributions are more accurate because they consider a broader context and hence are less sensitive to short-term fluctuations.

The Viterbi algorithm aims to find the single most probable sequence of states, by trying to maximize the joint probability of the observations and the states i.e., $p(x_1, ..., x_t, z_1, ..., z_t)$. It behaves the same way as the Smoothed distribution, in the sense that it accounts for the entire data while calculating the probability of the hidden state. However, the drawback and constraint with Viterbi algorithm is that it makes sure that the states in the probable path are continuous(no jumps) regardless of the probability distribution, while in the Smoothed distribution, we may find jumps from one state to non-neighboring states, because this states may be more probable. Because of this drawback and the noisy data, the probable path obtained from Viterbi algorithm is having lesser accuracy than the Smoothed distribution.

Question 6

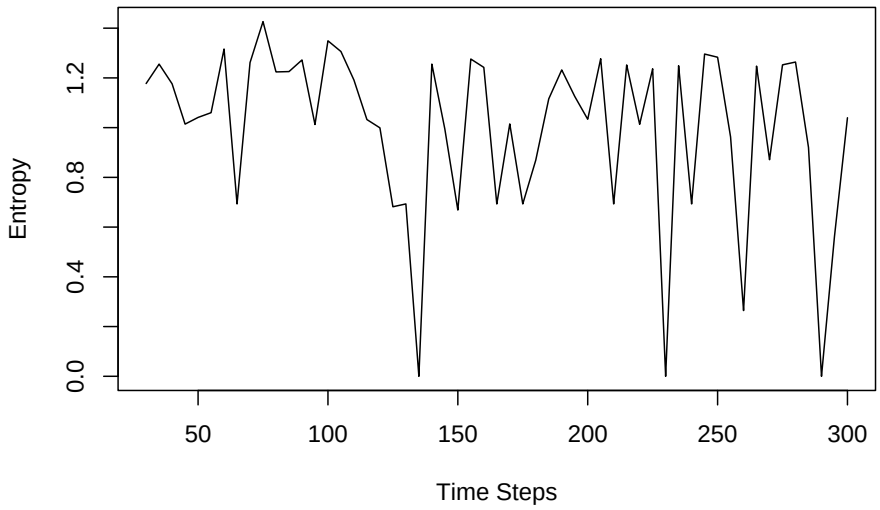
In general, one would expect that later in time, as the number of observations increase we know better about the robot's position.

To do so, we are checking the entropy of the filtered distribution. However, since we have faulty/noisy observations(recall that there is only 20% probability that a given observation is correct) there is bound to be considerable amount of uncertainty even as the number of observations increase.

```
entropies <- c()
new_simHMM <- simHMM(hmm = hmm_model1, length = 300)
new_obs <- new_simHMM$observation
Alpha_new <- exp(forward(hmm = hmm_model1, observation = new_obs))
filtered <- apply(Alpha_new,2,prop.table)
for (i in seq(30, 300, 5)) {

  entropies <- c(entropies, entropy.empirical(filtered[,i]))
}

plot(seq(30, 300, 5),entropies,
     type = 'l', xlab = 'Time Steps', ylab = 'Entropy')
```



Question 7

To compute the probability of hidden states for time step 101 given time step 100, we can write it as:

$$p(z^{T+1}|x^{1:T}) = \sum_{z^T} p(z^{T+1}, z^T|x^{1:T})$$

that is,

$$p(z^{101}|x^{1:100}) = \sum_{z^{100}} p(z^{101}, z^{100}|x^{1:100})$$

$$= \sum_{z^{100}} p(z^{100}|x^{1:100}) \cdot p(z^{101}|z^{100})$$

That is, the probability distribution of the hidden state at time step 101 is given by the product of the 100th time step of the filtered distribution and the transition matrix.

```
print('The probabilities of the hidden states of time step 101 given time step 100 is:')
```

```
## [1] "The probabilities of the hidden states of time step 101 given time step 100 is:"
```

```
transMat %*% sim100_results$filter_mat[,100]
```

```
##          [,1]
## S1  0.0000000
## S2  0.0000000
## S3  0.0000000
## S4  0.0000000
## S5  0.0000000
## S6  0.0000000
## S7  0.1424918
## S8  0.5000000
## S9  0.3575082
## S10 0.0000000
```

Contribution:

All the 3 members worked on all the questions, and then the solutions were compared. All 3 members collaborated in documenting the report. For Q1-2, Aswath's solution was used, for Q3-4 Akshath's solution was used, for Q5-7 Varun's solution was used. In addition, for Q6-7, we reasoned the solution as a group.