Lab 2 Individual Report

Akshay Gurudath

21/09/2021

Problem 1

Please note that all codes are attached in the appendix

Building a Hidden Markov Model for the Robot problem

Here, we can model the known position of the robot as the hidden state Z_t which is unknown. The next state $Z_{t+1}$is only dependent on the previous hidden state Z_t and this dependence can be modelled as Markovian. Now the position of the robot directly influences the measurement on the tracking device. Therefore, we can say that Z_t influences the observed state X_t. We can therefore model this as follows:

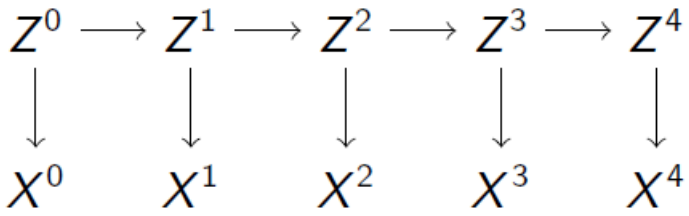


Figure 1: Hidden Markov Model

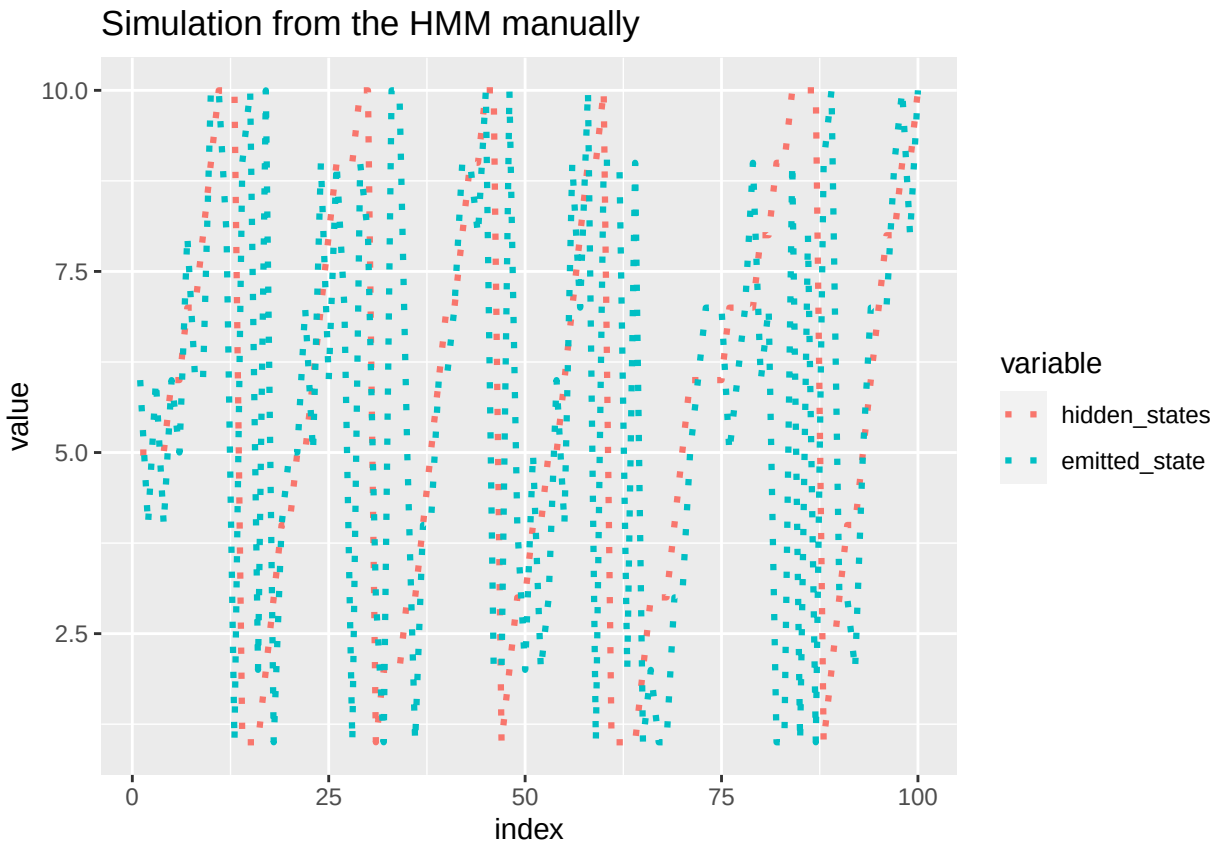
Problem 2

Simulating the HMM

There are 2 ways to simulate the HMM. The first way I have used is by simulating the HMM without the HMM package. This is relatively easy to do as the transition and emission dynamics are known.

```
initial<-ceiling(runif(1,0,10))

states=rep(0,100)
emitted_state=states
states[1]=initial

correction<-function(x){
  if(x<=0){
    return(10+x)
```

```
  } else if(x>10){
    return(x%%10)
  }else
    return(x)

}

for(i in 1:100){
  possible_emitted_states = c(states[i]-1,states[i]-2,states[i],(states[i]+1),(states[i]+2))
  #Possible emitted states from state i
  possible_emitted_states = sapply(possible_emitted_states,correction)
  #Correcting so that the range of states is 1-10

  emitted_state[i] = sample(possible_emitted_states,1)
  #Choosing a state at random from the possible emitted states
  if(i+1<=100){
  states[i+1]<-ifelse(runif(1)<0.5,states[i],(states[i]%%10+1))#State influencing the next state
  }
}



df<-melt(data.frame(cbind("index"=seq(1,100),"hidden_states"=states,emitted_state)),id.vars="index")

ggplot(df,aes(x = index, y = value, color = variable))+geom_line(linetype = 3,
            lwd = 1.1)+labs(title="Simulation from the HMM manually")
```



The second way is using the package HMM and specifically using the function simHMM. This is easier than the above method. However, we need to manually define the transition probability, initial probability and emission probability in order to define the HMM.

```
hidden_state=seq(1,10)
observed_state=seq(1,10)

start_probs=rep(1/10,10)


trans_prob=0.5*(diag(10)+diag(10)[c(seq(2,10),1),])

emission_prob = 0.2*(diag(10)[c(9,10,seq(1,8)),]+diag(10)[c(10,seq(1,9)),]+
                       diag(10)+diag(10)[c(seq(2,10),1),]+diag(10)[c(seq(3,10),1,2),])


kable(trans_prob,caption = "State Transition Probabilities")
```

Table 1: State Transition Probabilities

0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0	0.0

Table 1: State Transition Probabilities

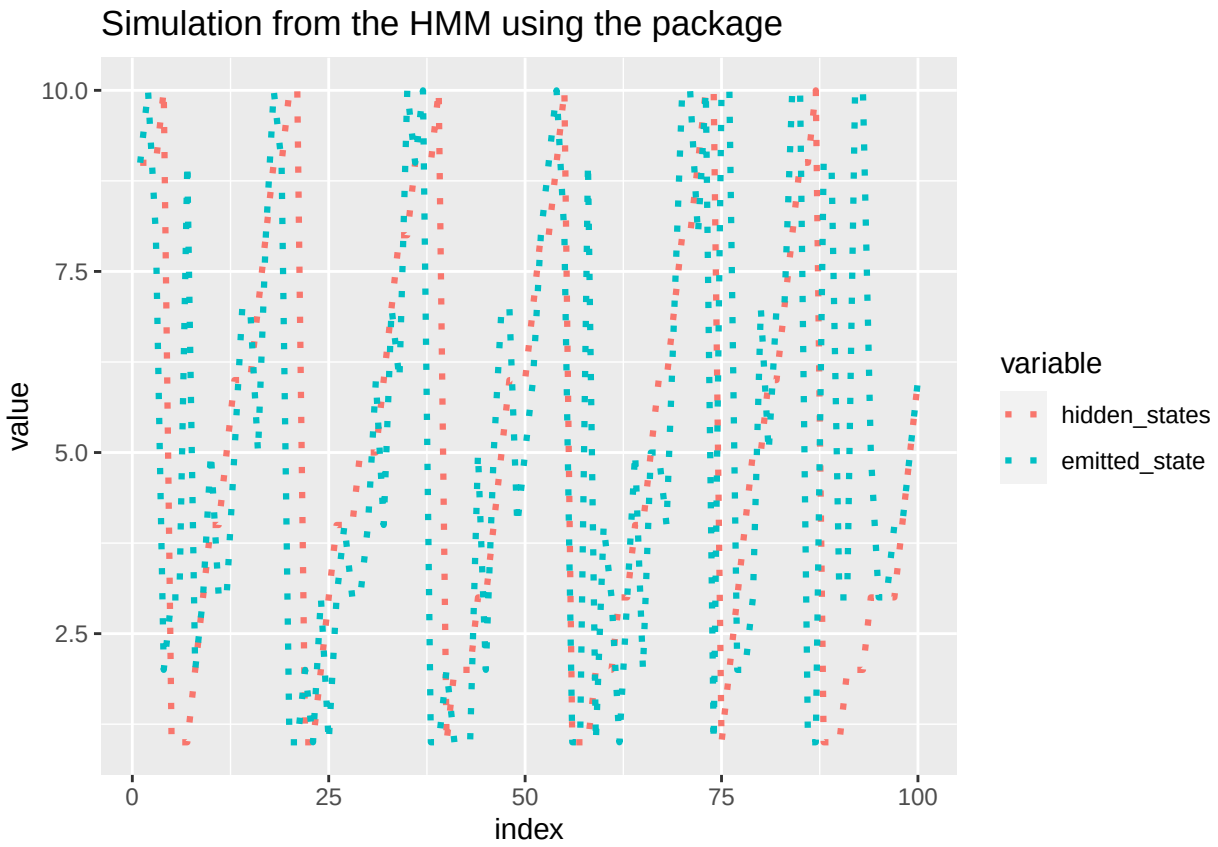
0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5
0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5

```
kable(emission_prob,caption ="State to Observation emission probabilities")
```

Table 2: State to Observation emission probabilities

0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2
0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2
0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0
0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0
0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0
0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.0
0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2
0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2
0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2

```
hmm=initHMM(States=hidden_state,Symbols=observed_state,startProbs = start_probs,transProbs = trans_prob



a=simHMM(hmm,length=100)

observation=a$observation
states=a$states


df<-melt(data.frame(cbind("index"=seq(1,100),"hidden_states"=states,"emitted_state"=observation)),id.va

ggplot(df,aes(x = index, y = value, color = variable))+geom_line(linetype = 3,
            lwd = 1.1)+labs(title="Simulation from the HMM using the package")
```



With both these methods we see similar graphs. The second method used to instantiate the HMM will prove useful for finding out the forward, backward distribution and the most probable path.

Problem 3 and 4

Filter, Smoothing distributions and the most probable paths

Please note that I have combined problems 3 and 4 in the same function. The distribution needs to be computed and then alone can we compute the accuracies. If needed, the distributions can also be returned

We will use the HMM model instantiated above as well as the observations obtained through simulation.

The smoothing distribution is given by $p(z_t|x^{0:T}) = \frac{\alpha(z^t)\beta(z^t)}{\sum_{z^t}\alpha(z^t)\beta(z^t)}$

The filter distribution is given by $p(z_t|x^{0:t}) = \frac{\alpha(z^t)}{\sum_{z^t}\alpha(z^t)}$

Both these distributions are obtained by first obtaining $\alpha(z^t)$ and $\beta(z^t)$ using the forward and backward functions in HMM. The most probable path is obtained by simply running the viterbi function.

The below function calculates the forward, backward distribution along with the most probable path.

In addition it returns the accuracy of all these three paths when compared to the actual hidden states.

```
#Input the HMM and the observations along with the actual states. The actual states are used for compar
hidden_markov<-function(hmm,observation,states){
```

```
  backward_hmm=exp(backward(hmm,observation))
  forward_hmm=exp(forward(hmm,observation))

  filter_distribution=prop.table(forward_hmm,margin = 2)

  smoothing_distribution=prop.table(backward_hmm*forward_hmm,margin=2)

  most_probable_path=viterbi(hmm,observation)

  guessed_forward=apply(filter_distribution,MARGIN=2,FUN=which.max)

  guessed_backward=apply(smoothing_distribution,MARGIN=2,FUN=which.max)
  acc<-list(rep(0,3))
  acc[1]=table(states==guessed_forward)[[2]]/sum(table(states==guessed_forward))
  acc[2]=table(states==guessed_backward)[[2]]/sum(table(states==guessed_backward))
  acc[3]=table(states==most_probable_path)[[2]]/sum(table(states==most_probable_path))
  names(acc)<-c("Filter Distribution","Smoothing Distribution", "Most Probable Path")

  return(acc)
}

acc=hidden_markov(hmm,observation,states)
print("Below is the accuracy of the path for the three methods ")
```

```
## [1] "Below is the accuracy of the path for the three methods "
```

```
print(acc)
```

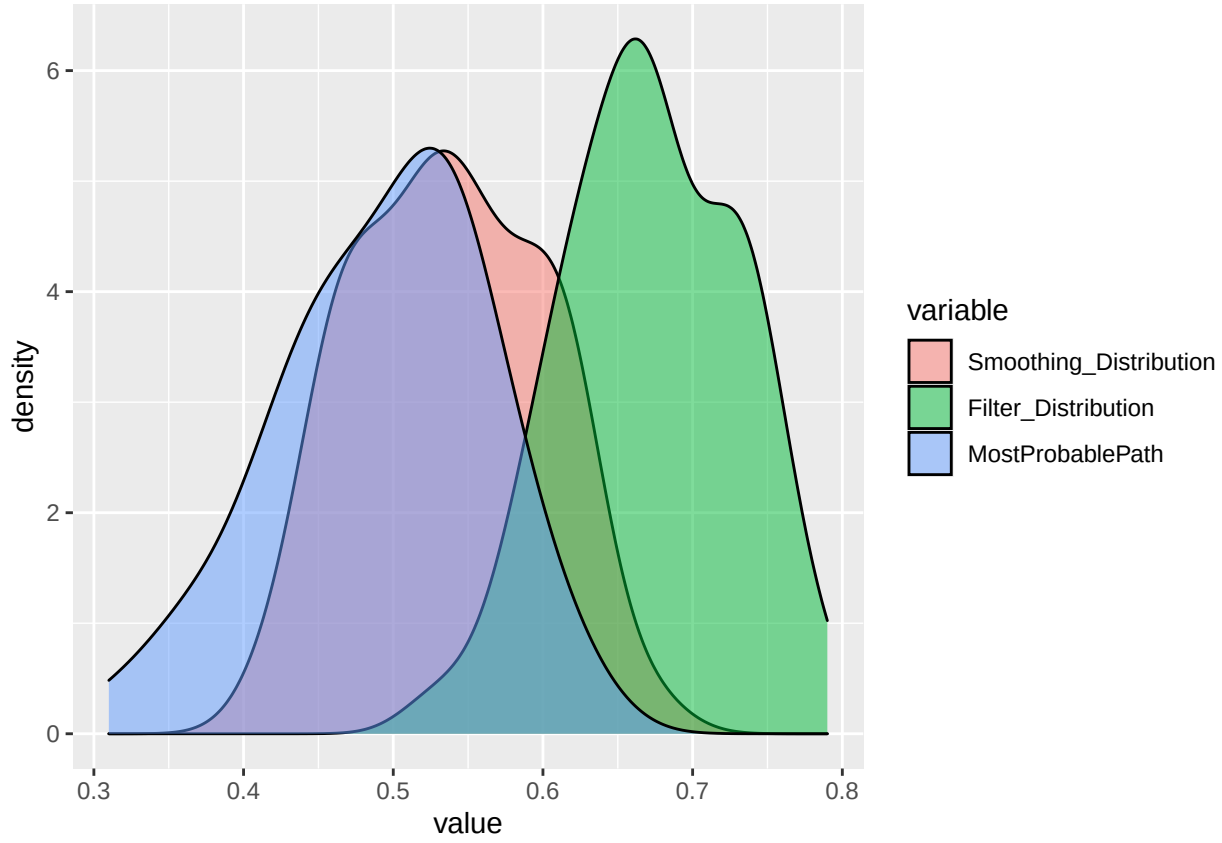
```
## $`Filter Distribution`
## [1] 0.57
##
## $`Smoothing Distribution`
## [1] 0.7
##
## $`Most Probable Path`
## [1] 0.44
```

Problem 5

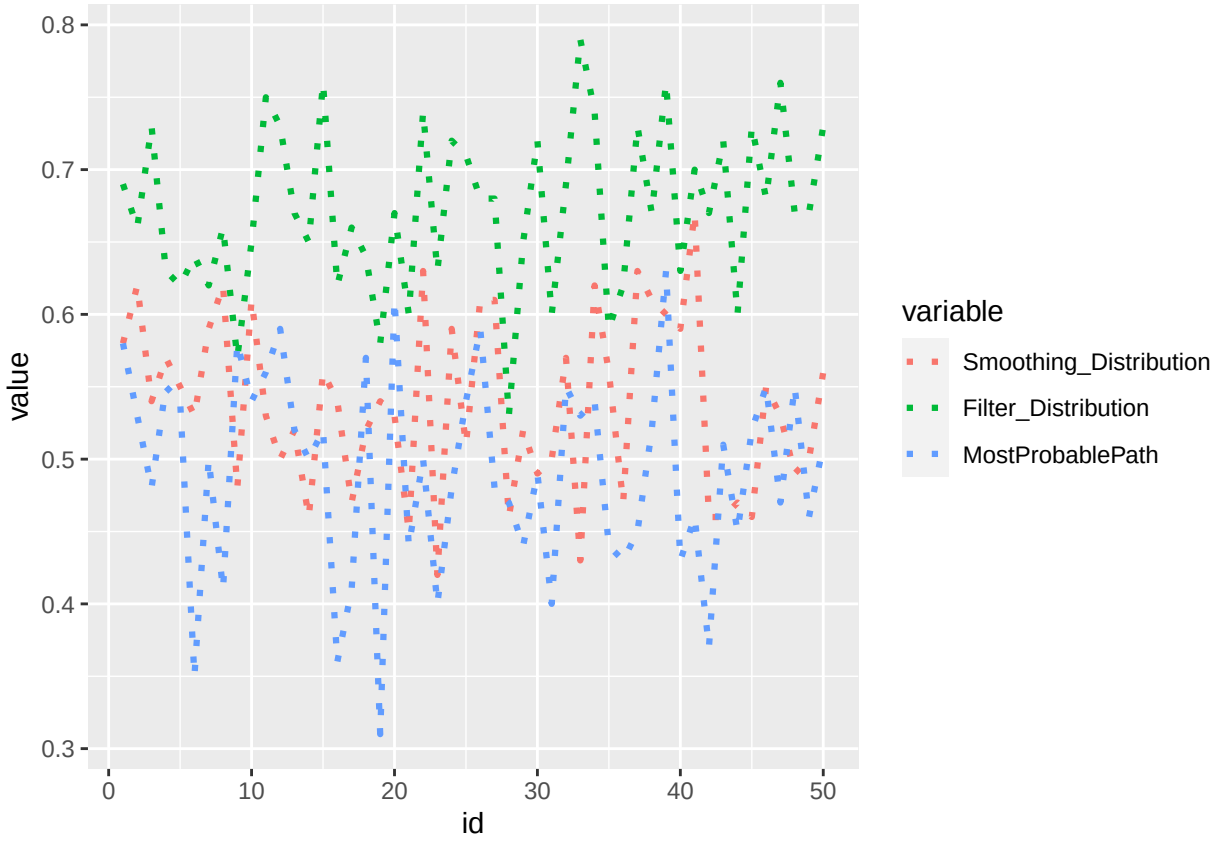
Many Simulated Samples to check accuracy of all three methods

```
n_samples=50


accuracy<-data.frame(cbind("Smoothing_Distribution"=rep(0,n_samples),"Filter_Distribution"=rep(0,n_sampl

for(i in 1:n_samples){
a=simHMM(hmm,length=100)
```

```
accuracy[i,]<-hidden_markov(hmm,a$observation,a$states)
}
accuracy$id=1:n_samples
df<-melt(accuracy,id.vars = "id")

ggplot(df,aes(x=value,fill=variable))+geom_density(alpha=0.5)
```




```
ggplot(df,aes(x=id,y=value,color=variable))+geom_line(linetype = 3,
            lwd = 1.1)
```



From here, we see that the accuracy obtained from smoothing distribution seems to be higher than both the filter distribution and the most probable path. The accuracy is higher than the filter distribution because a smoothing distribution takes into account *all of the observations* for estimating the distribution whereas the filter distribution only takes into account *the observation till then.*

Next, the accuracy of the smoothing distribution is higher than the accuracy of the most probable path as most probable path maximizes over all possible paths $Z_{0:T}$ whereas the smoothing distribution just maximizes over Z_t

$z_{0:t} = \arg\max p(z^{0:T}|x^{0:T})$ for the most probable path And $z_t = \arg\max p(z_t|x^{0:T})$ for the smoothing distribution

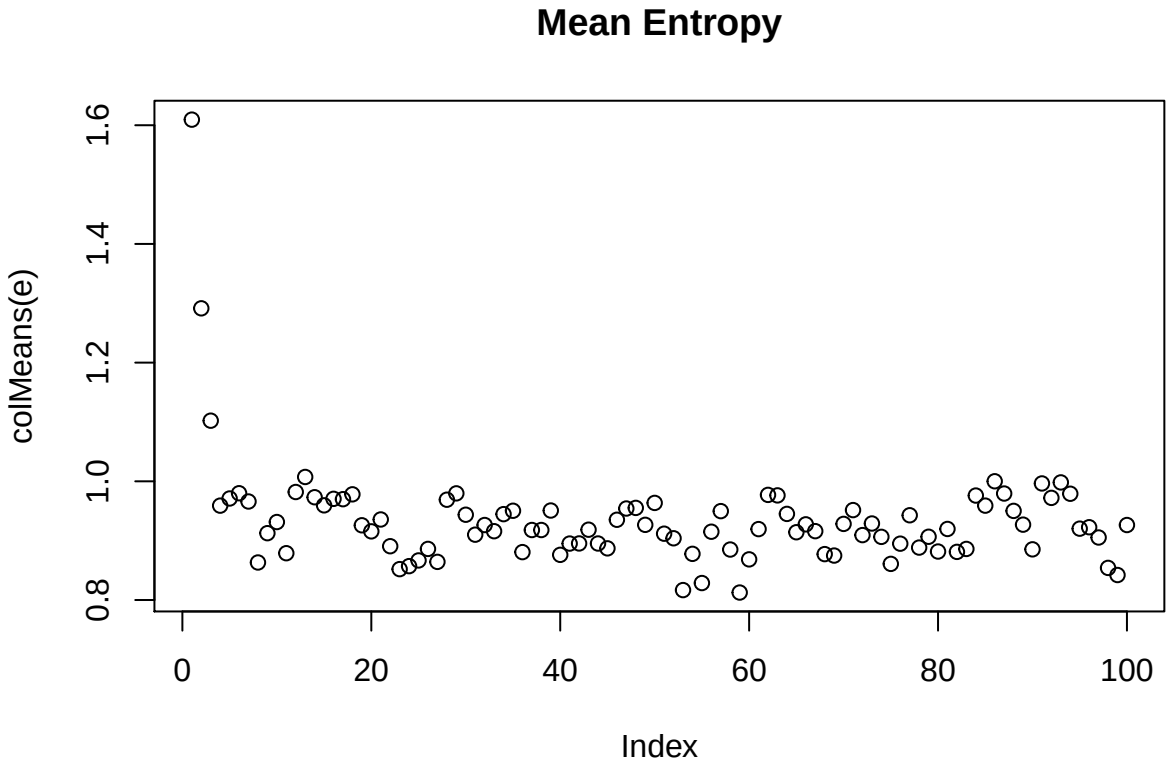
The smoothing distribution may give paths that don't follow the state transitions but the most probable path will always give us a path that can follow the state transitions. Because of this constraint that it has to give a path that makes sense, overall accuracy may be lower but the sequence obtained from the most probable path will make more sense.

Problem 6

Entropy of the filter distribution with time

```
entropy_cal<-function(hmm,observation){
  backward_hmm=exp(backward(hmm,observation))
  forward_hmm=exp(forward(hmm,observation))
```

```
  filter_distribution=prop.table(forward_hmm,margin = 2)
  e=rep(0,100)

  for(i in 1:100){
  e[i]=entropy.empirical(filter_distribution[,i])
  }

  return(e)
}
e=matrix(nrow=50,ncol=100)

for(i in 1:50){
  e[i,]=entropy_cal(hmm,simHMM(hmm,100)$observation)

}

plot(colMeans(e),main="Mean Entropy")
```



No this does not seem the case as there is no clear decrease in entropy.

Problem 7

Prediction Problem

$p(z^{t+1}|x^{0:t}) = \sum_{z_t} p(z^{t+1}, z^t|x^{0:t}) = \sum_{z_t} p(z^{t+1}|z^t, x^{0:t})p(z^t|x^{0:t})$

Now we know that $p(z^{t+1}|z^t, x^{0:t}) = p(z^{t+1}|z^t)$

Therefore, $p(z^{t+1}|x^{0:t}) = \sum_{z_t} p(z^{t+1}|z^t)p(z^t|x^{0:t})$ where $p(z^t|x^{0:t})$ is a filter distribution and $p(z^{t+1}|z^t)$ is the transition matrix.

Thus this prediction distribution can easily be computed.

```
observation=simHMM(hmm,100)$observation

backward_hmm=exp(backward(hmm,observation))
forward_hmm=exp(forward(hmm,observation))

filter_distribution=prop.table(forward_hmm,margin = 2)

smoothing_distribution=prop.table(backward_hmm*forward_hmm,margin=2)

pred_distribution=filter_distribution[,100]%*%trans_prob

print(pred_distribution)
```

```
##             [,1] [,2] [,3] [,4] [,5]       [,6] [,7]      [,8]      [,9]
## [1,] 0.02307692    0    0    0    0 0.07384615 0.22 0.3076923 0.2569231
##           [,10]
## [1,] 0.1184615
```

```
print("Shubham")
```

```
## [1] "Shubham"
```