Lab2

Yifan Ding, Chao Fu, Yunan Dong, Ravinder Reddy

25 September, 2021

Q1.1

```r
Trans <- matrix(0, nrow = 10, ncol = 10)
Emiss <- matrix(0, nrow = 10, ncol = 10)

re_index <- function(i){
  i <- i %% 10
  if (i %% 10 == 0) {
    i = 10
  }
  return(i)
}

for (i in 1:10) {

  Trans[i, i] = 0.5
  Trans[i, re_index(i+1)] = 0.5

  Emiss[i, i] <- 0.2
  Emiss[i, re_index(i+1)] <- 0.2
  Emiss[i, re_index(i-1)] <- 0.2
  Emiss[i, re_index(i+2)] <- 0.2
  Emiss[i, re_index(i-2)] <- 0.2
}


model <- initHMM(States = c(1:10), Symbols = c(1:10),
               transProbs = Trans,
               emissionProbs = Emiss)
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
```

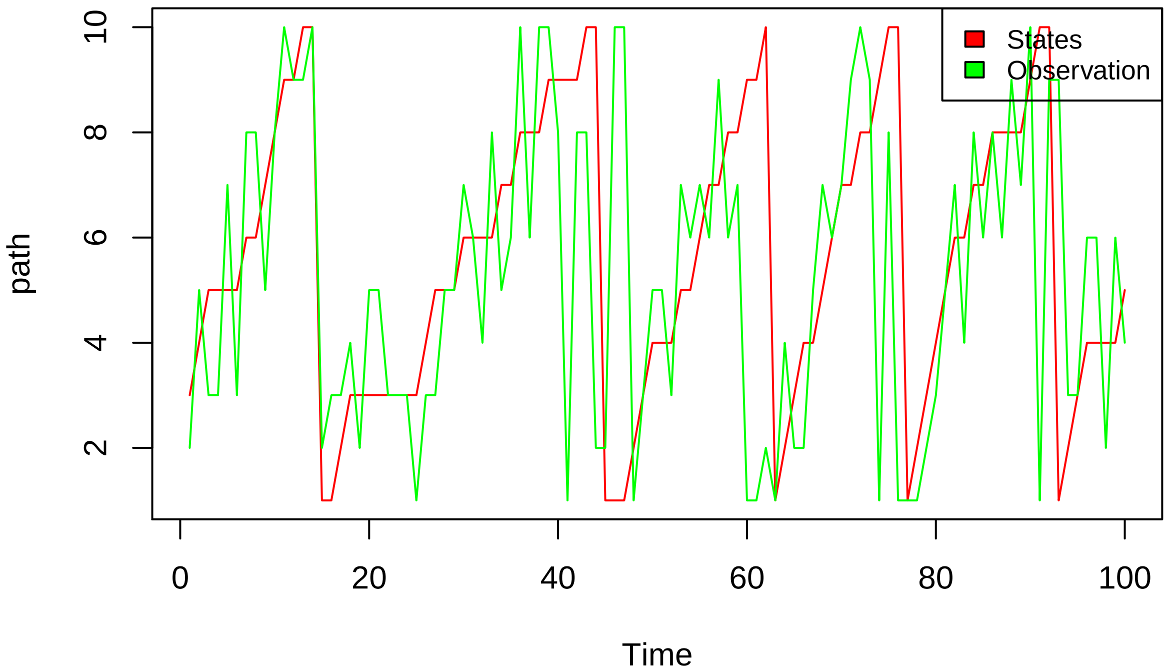
```
##     to
## from   1   2   3   4   5   6   7   8   9  10
##   1  0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
##   2  0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0
##   3  0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0
##   4  0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0
##   5  0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0
##   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
```

Q1.2

```
set.seed(1234)
samples <- simHMM(model, 100)
samples
```

```
## $states
##   [1]  3  4  5  5  5  5  6  6  7  8  9  9 10 10  1  1  2  3  3  3  3  3  3  3  3
##  [26]  4  5  5  5  6  6  6  6  7  7  8  8  8  9  9  9  9 10 10  1  1  1  2  3  4
##  [51]  4  4  5  5  6  7  7  8  8  9  9 10  1  2  3  4  4  5  6  7  7  8  8  9 10
##  [76] 10  1  2  3  4  5  6  6  7  7  8  8  8  8  9 10 10  1  2  3  4  4  4  4  5
##
## $observation
##   [1]  2  5  3  3  7  3  8  8  5  8 10  9  9 10  2  3  3  4  2  5  5  3  3  3  1
##  [26]  3  3  5  5  7  6  4  8  5  6 10  6 10 10  8  1  8  8  2  2 10 10  1  3  5
##  [51]  5  3  7  6  7  6  9  6  7  1  1  2  1  4  2  2  5  7  6  7  9 10  9  1  8
##  [76]  1  1  1  2  3  5  7  4  8  6  8  6  9  7 10  1  9  9  3  3  6  6  2  6  4
```

```
plot(samples$states, type='l', col='red', xlab = 'Time', ylab = 'path')
lines(samples$observation, col='green')
legend("topright", legend = c("States", "Observation"),
         fill = c("red", "green"), cex = 0.8)
```



Q1.3

```
fw <- exp(forward(model, samples$observation))
bw <- exp(backward(model, samples$observation))

filtering <- prop.table(fw, 2)
smoothing <- prop.table(fw*bw, 2)
```

Q1.4

```
filter_postion = apply(filtering, 2, which.max)
smooth_postion = apply(smoothing, 2, which.max)
viterbi_position<- viterbi(model, samples$observation)

acc_filter <- sum(filter_postion == samples$states) / 100
print(paste("Accuracy of filter:",acc_filter) )
```

```
## [1] "Accuracy of filter: 0.63"
```

```
acc_smooth <- sum(smooth_postion == samples$states) / 100
print(paste("Accuracy of smooth:",acc_smooth) )
```

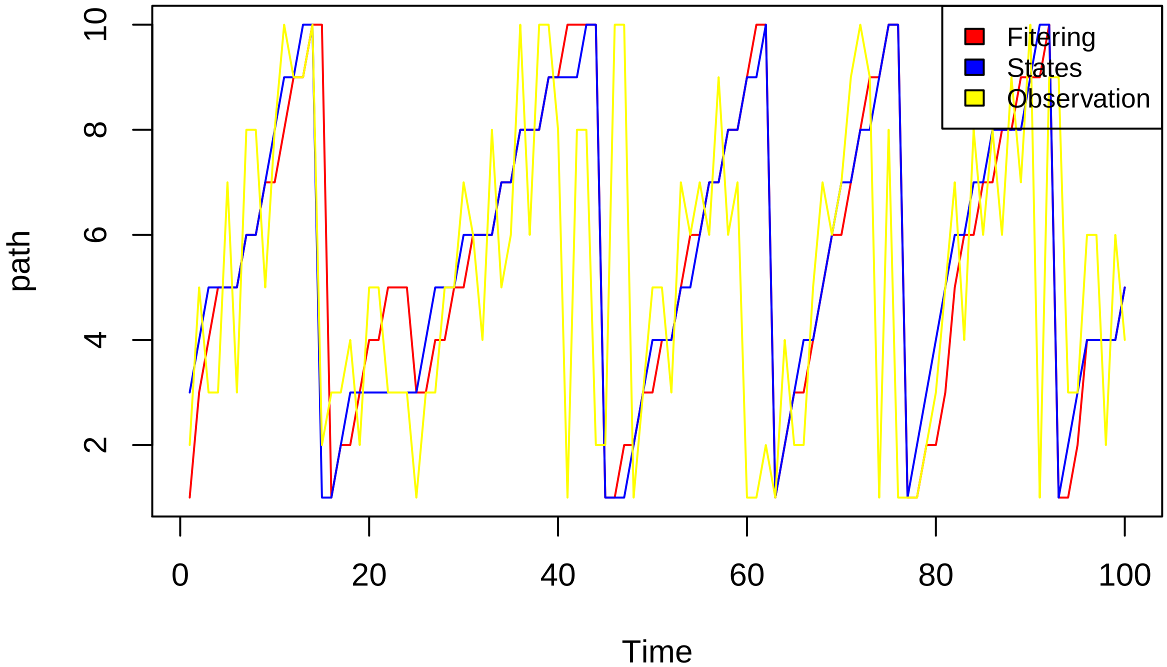
```
## [1] "Accuracy of smooth: 0.75"
```

```
acc_viterbi <- sum(viterbi_position == samples$states) / 100
print(paste("Accuracy of viterbi:",acc_viterbi) )
```

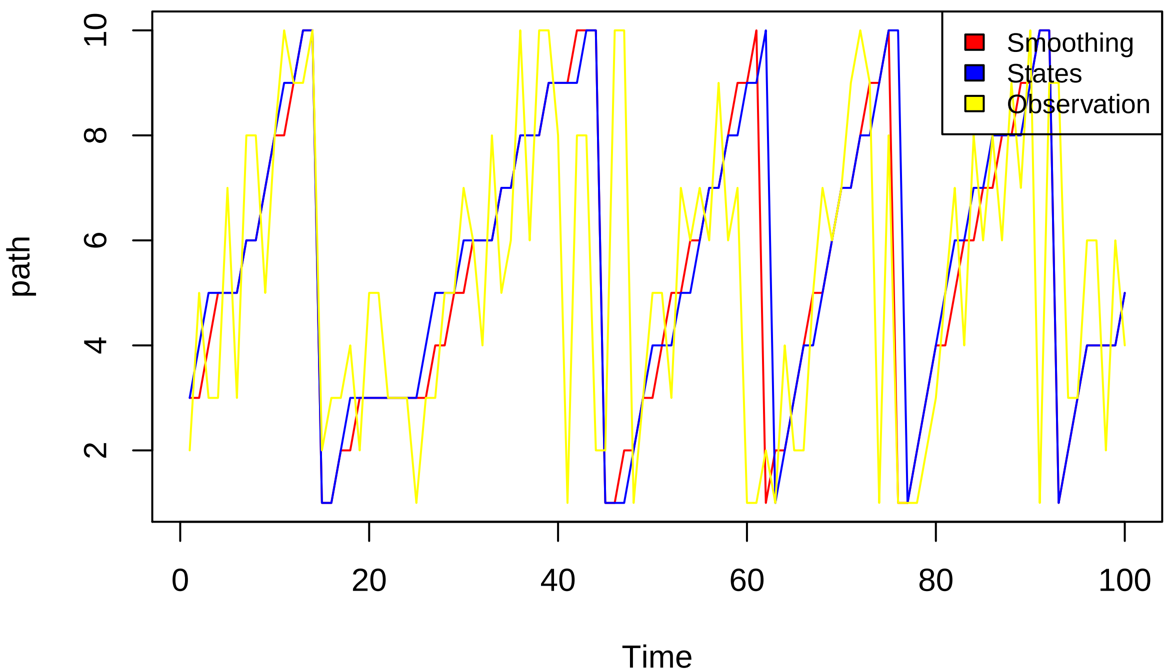
```
## [1] "Accuracy of viterbi: 0.49"
```

```
plot(filter_postion, type='l', col='red', xlab = 'Time', ylab = 'path')
lines(samples$states, col='blue')
lines(samples$observation, col='yellow')
legend("topright", legend = c("Fitering", "States", "Observation"),
```

```
        fill = c("red", "blue", "yellow"), cex = 0.8)
```

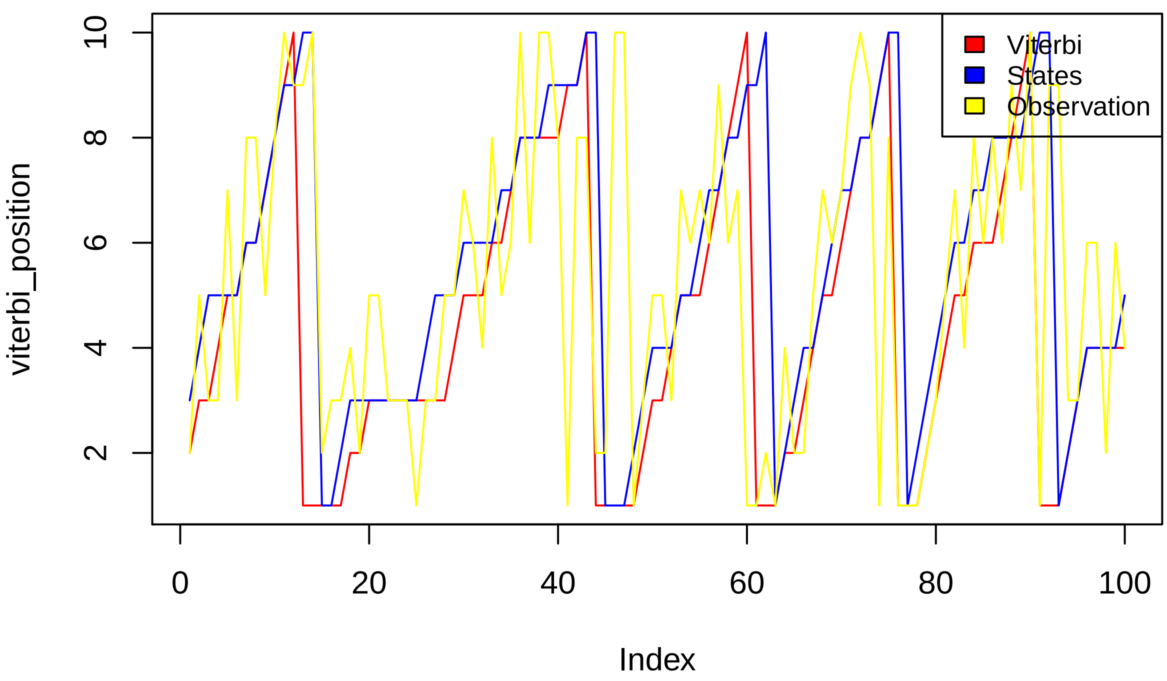



```
plot(smooth_postion, type='l', col='red', xlab = 'Time', ylab = 'path')
lines(samples$states, col='blue')
lines(samples$observation, col='yellow')
legend("topright", legend = c("Smoothing", "States", "Observation"),
        fill = c("red", "blue", "yellow"), cex = 0.8)
```




```
plot(viterbi_position, type='l', col='red')
lines(samples$states, col='blue')
lines(samples$observation, col='yellow')
```

```r
legend("topright", legend = c("Viterbi", "States", "Observation"),
       fill = c("red", "blue", "yellow"), cex = 0.8)
```



Q1.5

```r
set.seed(1234)
repeat_size <- 100
acc_filters <- c()
acc_smooths <- c()
acc_viterbis <- c()

for (i in 1:repeat_size) {

  sample_size <- 100
  samples <- simHMM(model, sample_size)
  fw <- exp(forward(model, samples$observation))
  bw <- exp(backward(model, samples$observation))

  filtering <- prop.table(fw, 2)
  smoothing <- prop.table(fw*bw, 2)

  filter_postion = apply(filtering, 2, which.max)
  smooth_postion = apply(smoothing, 2, which.max)
  viterbi_position<- viterbi(model, samples$observation)

  acc_filter <- sum(filter_postion == samples$states) / sample_size
  acc_smooth <- sum(smooth_postion == samples$states) / sample_size
  acc_viterbi <- sum(viterbi_position == samples$states) / sample_size

  acc_filters <- c(acc_filters, acc_filter)
  acc_smooths <- c(acc_smooths, acc_smooth)
```

```
  acc_viterbis <- c(acc_viterbis, acc_viterbi)
}

avg_acc_filter <- mean(acc_filters)
print(paste("Average accuracy of filtering:",avg_acc_filter) )
```

```
## [1] "Average accuracy of filtering: 0.5417"
```

```
avg_acc_smooth <- mean(acc_smooths)
print(paste("Average accuracy of smoothing:",avg_acc_smooth) )
```

```
## [1] "Average accuracy of smoothing: 0.6824"
```

```
avg_acc_viterbi <- mean(acc_viterbis)
print(paste("Average accuracy of viterbi:",avg_acc_viterbi) )
```

```
## [1] "Average accuracy of viterbi: 0.5111"
```

In general, the smoothed distributions should be more accurate than the filtered distributions. Why ? In general, the smoothed distributions should be more accurate than the most probable paths, too. Why ?

The filtered distribution is given below:

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

The smoothing distribution is given below:

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

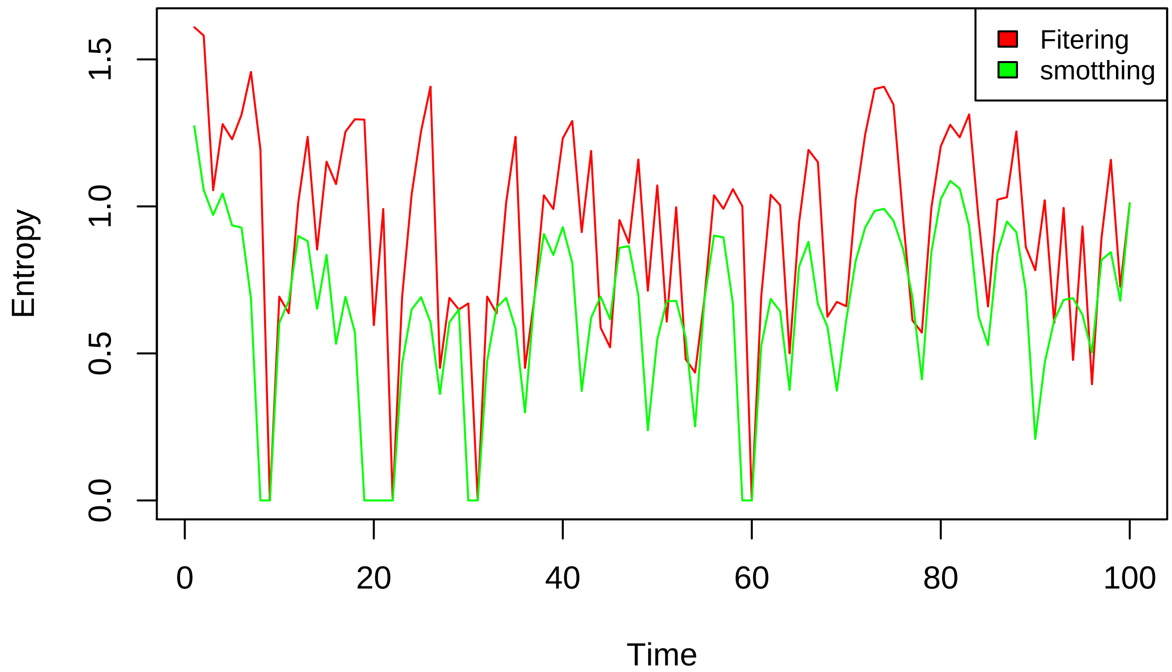
The most probable path:

$$Z^{0:T}_{max} = argmax_{z^{0:T}} P(Z^{0:T}|X^{0:T})$$

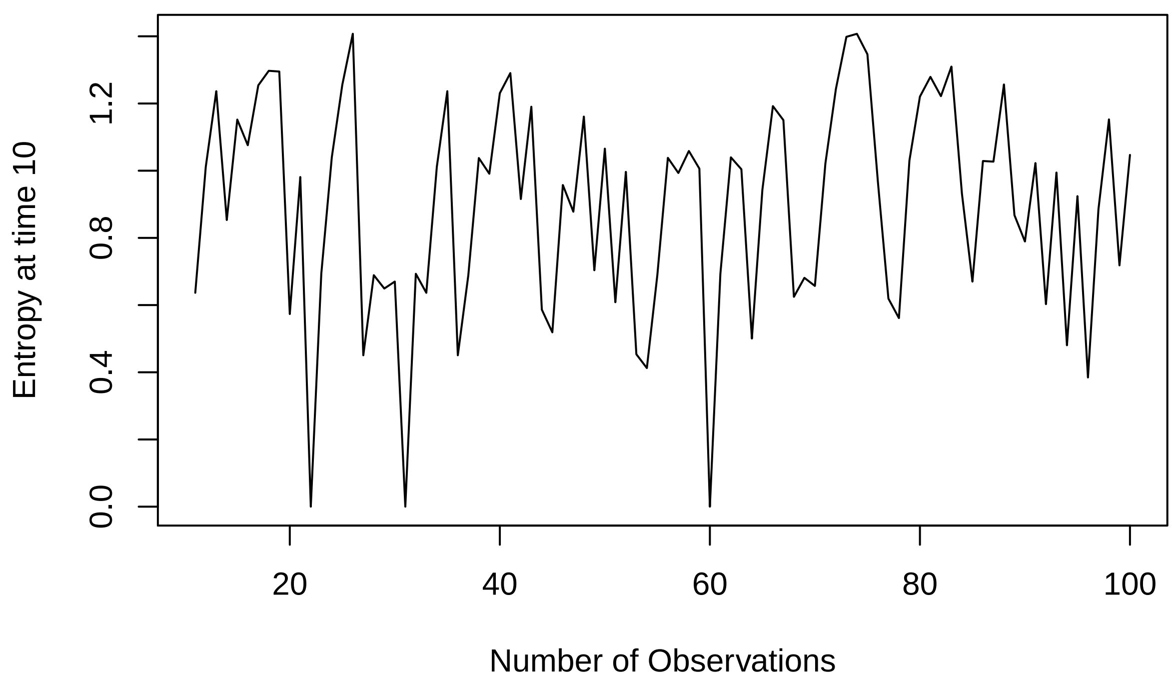
Answer: Smoothed distribution is conditioning on all observations, while filtered distribution is only conditioning on up-to-date observations. Later observations should not effect previous state only if we know the exact physics model and correct observations, but here HMM is probabilistic estimation(we dont have exact physics model as well as correct observations). Therefore HMM is doing probabilistic estimation by Bayes theorem, but we believe state at time t+1, t+2…t+n are dependent on state at time t, therefore the posterior on more observations can be helpful in smoothing. For most probable path, this approach try to maximize joint probability of hidden state $P(Z^{0:T}|X^{0:T}$, which can not guarantee the best solution of each hidden state $P(z^t|X^{0:T})$.

Q1.6

```
entropy_filter <- lapply(c(1:100),function(i)entropy.empirical(filtering[,i]))
entropy_smooth <- lapply(c(1:100),function(i)entropy.empirical(smoothing[,i]))
plot(unlist(entropy_filter), type='l', col='red', xlab = 'Time', ylab = 'Entropy')
lines(unlist(entropy_smooth), col='green')
legend("topright", legend = c("Fitering", "smotthing"),
       fill = c("red", "green"), cex = 0.8)
```




```r
Hs <- c()
for (i in 1:90) {
  fw <- exp(forward(model, samples$observation[1:10+i]))
  bw <- exp(backward(model, samples$observation[1:10+i]))
  posterior(model, samples$observation[1:10+i])
  smoothing <- prop.table(fw*bw, 2)
  H <- entropy.empirical(smoothing[ ,10])
  Hs <- c(Hs, H)
}
plot(c(11:100),Hs, type = 'l', xlab = 'Number of Observations', ylab = 'Entropy at time 10')
```



$$H(x) = -\sum_{n=i}^{N} p_i(x) log p_i(x)$$

According to Information theory, higher entropy indicate higher uncertainty, (i.e. an event happens with probability 1, means no uncertainty, entropy is 0), so the lower entropy the more confidence. Clearly, smoothing has lower entropy comparing filtering in most time point. But we also checked the trend of Entropy for a single time point at 10 with an increasing number of observations, we find the trend is quite flat, seems indicate entropy(uncertainty) doesn't decrease with more observations.

```
Z_100= as.matrix(filtering[,100])
Z_100
```

```
##         [,1]
## 1  0.0000000
## 2  0.0000000
## 3  0.0000000
## 4  0.0000000
## 5  0.0000000
## 6  0.0000000
## 7  0.3459091
## 8  0.4922424
## 9  0.1618484
## 10 0.0000000
```

```
Z_101 <- t(model$transProbs)%*%Z_100

print(paste("State 101 most likely be", which.max(Z_101)))
```

```
## [1] "State 101 most likely be 8"
```

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(HMM)
library(entropy)
Trans <- matrix(0, nrow = 10, ncol = 10)
Emiss <- matrix(0, nrow = 10, ncol = 10)

re_index <- function(i){
  i <- i %% 10
  if (i %% 10 == 0) {
    i = 10
  }
  return(i)
}

for (i in 1:10) {

  Trans[i, i] = 0.5
  Trans[i, re_index(i+1)] = 0.5

  Emiss[i, i] <- 0.2
  Emiss[i, re_index(i+1)] <- 0.2
  Emiss[i, re_index(i-1)] <- 0.2
```

```r
  Emiss[i, re_index(i+2)] <- 0.2
  Emiss[i, re_index(i-2)] <- 0.2
}


model <- initHMM(States = c(1:10), Symbols = c(1:10),
               transProbs = Trans,
               emissionProbs = Emiss)
model
set.seed(1234)
samples <- simHMM(model, 100)
samples
plot(samples$states, type='l', col='red', xlab = 'Time', ylab = 'path')
lines(samples$observation, col='green')
legend("topright", legend = c("States", "Observation"),
         fill = c("red", "green"), cex = 0.8)
fw <- exp(forward(model, samples$observation))
bw <- exp(backward(model, samples$observation))

filtering <- prop.table(fw, 2)
smoothing <- prop.table(fw*bw, 2)
filter_postion = apply(filtering, 2, which.max)
smooth_postion = apply(smoothing, 2, which.max)
viterbi_position<- viterbi(model, samples$observation)

acc_filter <- sum(filter_postion == samples$states) / 100
print(paste("Accuracy of filter:",acc_filter) )
acc_smooth <- sum(smooth_postion == samples$states) / 100
print(paste("Accuracy of smooth:",acc_smooth) )
acc_viterbi <- sum(viterbi_position == samples$states) / 100
print(paste("Accuracy of viterbi:",acc_viterbi) )

plot(filter_postion, type='l', col='red', xlab = 'Time', ylab = 'path')
lines(samples$states, col='blue')
lines(samples$observation, col='yellow')
legend("topright", legend = c("Fitering", "States", "Observation"),
         fill = c("red", "blue", "yellow"), cex = 0.8)


plot(smooth_postion, type='l', col='red', xlab = 'Time', ylab = 'path')
lines(samples$states, col='blue')
lines(samples$observation, col='yellow')
legend("topright", legend = c("Smoothing", "States", "Observation"),
         fill = c("red", "blue", "yellow"), cex = 0.8)

plot(viterbi_position, type='l', col='red')
lines(samples$states, col='blue')
lines(samples$observation, col='yellow')
legend("topright", legend = c("Viterbi", "States", "Observation"),
         fill = c("red", "blue", "yellow"), cex = 0.8)


set.seed(1234)
```

```r
repeat_size <- 100
acc_filters <- c()
acc_smooths <- c()
acc_viterbis <- c()

for (i in 1:repeat_size) {

  sample_size <- 100
  samples <- simHMM(model, sample_size)
  fw <- exp(forward(model, samples$observation))
  bw <- exp(backward(model, samples$observation))

  filtering <- prop.table(fw, 2)
  smoothing <- prop.table(fw*bw, 2)

  filter_postion = apply(filtering, 2, which.max)
  smooth_postion = apply(smoothing, 2, which.max)
  viterbi_position<- viterbi(model, samples$observation)

  acc_filter <- sum(filter_postion == samples$states) / sample_size
  acc_smooth <- sum(smooth_postion == samples$states) / sample_size
  acc_viterbi <- sum(viterbi_position == samples$states) / sample_size

  acc_filters <- c(acc_filters, acc_filter)
  acc_smooths <- c(acc_smooths, acc_smooth)
  acc_viterbis <- c(acc_viterbis, acc_viterbi)
}

avg_acc_filter <- mean(acc_filters)
print(paste("Average accuracy of filtering:",avg_acc_filter) )
avg_acc_smooth <- mean(acc_smooths)
print(paste("Average accuracy of smoothing:",avg_acc_smooth) )
avg_acc_viterbi <- mean(acc_viterbis)
print(paste("Average accuracy of viterbi:",avg_acc_viterbi) )

entropy_filter <- lapply(c(1:100),function(i)entropy.empirical(filtering[,i]))
entropy_smooth <- lapply(c(1:100),function(i)entropy.empirical(smoothing[,i]))
plot(unlist(entropy_filter), type='l', col='red', xlab = 'Time', ylab = 'Entropy')
lines(unlist(entropy_smooth), col='green')
legend("topright", legend = c("Fitering", "smotthing"),
         fill = c("red", "green"), cex = 0.8)
Hs <- c()
for (i in 1:90) {
  fw <- exp(forward(model, samples$observation[1:10+i]))
  bw <- exp(backward(model, samples$observation[1:10+i]))
  posterior(model, samples$observation[1:10+i])
  smoothing <- prop.table(fw*bw, 2)
  H <- entropy.empirical(smoothing[ ,10])
  Hs <- c(Hs, H)
}
plot(c(11:100),Hs, type = 'l', xlab = 'Number of Observations', ylab = 'Entropy at time 10')

Z_100= as.matrix(filtering[,100])
```

```r
Z_100
Z_101 <- t(model$transProbs)%*%Z_100

print(paste("State 101 most likely be", which.max(Z_101)))
```