Lab3

Elvin Granat

2018-10-04

## Sample function from true hidden state z

Picks a distribution from the mixed and samples from it

z\_transition\_sample = function(z\_prev) {  
 pick = sample(1:3, 1)  
 switch(pick,  
 "first" = {  
 z = rnorm(1, z\_prev, 1)  
 },   
 "second" = {  
 z = rnorm(1, z\_prev + 1, 1)  
 },  
 "third" = {  
 z = rnorm(1, z\_prev + 2, 1)  
 }  
 )  
 return(z)  
}

## Sample function from the observed state x

Picks a distribution from the mixed and samples from it with given z as mean

x\_emission\_sample = function(z, sigma) {  
 pick = sample(1:3, 1)  
 switch(pick,  
 "first" = {  
 x = rnorm(1, z, 1)  
 },   
 "second" = {  
 x = rnorm(1, z - 1, 1)  
 },  
 "third" = {  
 x = rnorm(1, z + 1, 1)  
 }  
 )  
 return(x)  
}

## Simulation

Simulation of given SSM for T = 100 time steps

sim = function(sigma) {  
 T = 100  
 z = c() # true states  
 x = c() # observed states  
 for (t in 1:T) {  
 # z  
 if (t == 1) z[t] = runif(1, 0, 100)  
 else z[t] = z\_transition\_sample(z[t-1])  
   
 # x  
 x[t] = x\_emission\_sample(z[t], sigma)  
 }  
 return (data.frame(x, z))  
}

## Calculates the weights for each particle using known observation

In a way it describes the relative probabity of z being the true hidden value when x was observed

w\_density = function(x, z, sigma) {  
 w = c()  
 M = length(x)  
 for(m in 1:M) {  
 w1 = dnorm(x, mean = z, sd = sigma, log = FALSE)  
 w2 = dnorm(x, mean = z - 1, sd = sigma, log = FALSE)  
 w3 = dnorm(x, mean = z + 1, sd = sigma, log = FALSE)  
 w[m] = (w1 + w2 + w3) / 3  
 }  
 return(w)  
}

## Draws particles of given particles z with corresponding weights w

Utilizes multinomial distrubtion draw

particle\_draw = function(z\_particles, w) {  
 M = length(w)  
 res = c()  
 draws = rmultinom(1, M, w)  
 for (i in 1:M) {  
 if (draws[i]) {  
 for (j in 1:draws[i]) {  
 res = append(res, z\_particles[i])  
 }  
 }  
 }  
 return(res)  
}

## Particle filtering

Performs particle filtering according to algorithm on lecture 8

particle\_filtering = function(sim, sigma, equalWeight) {  
 x = sim$x # extract observations  
   
 T = 100  
 M = 100  
 Z = matrix(0, nrow = M, ncol = T)  
 Zbar = matrix(0, nrow = M, ncol = T)  
   
 for (t in 1:T) {  
 if (t == 1) Z[,1] = runif(M, 0, 100)  
 else {  
 z\_particles = c()  
 w = c()  
 for (m in 1:M) {  
 z\_particles[m] = z\_transition\_sample(Z[m,t-1])  
   
 if (equalWeight) w[m] = 1  
 else w[m] = w\_density(x[t], z\_particles[m], sigma)  
 }  
 Zbar[,t] = z\_particles  
 Z[,t] = particle\_draw(Zbar[,t], w)  
 }  
 }  
 return(Z)  
}

## Execution of simulation + particle filtering + visualization

A function for execution of the tasks

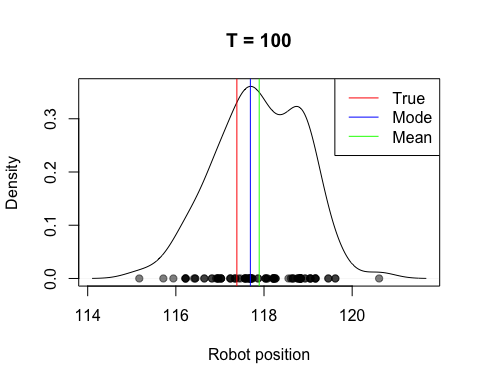
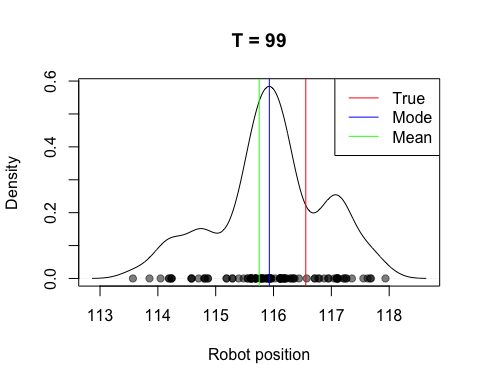
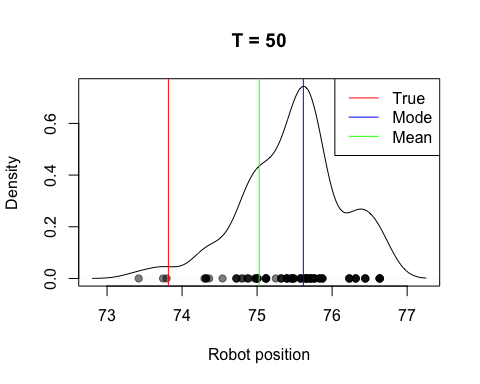
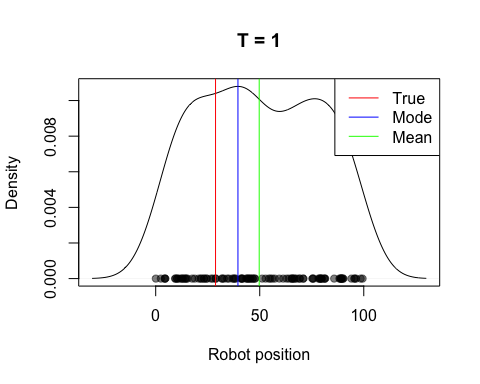
execute = function(sigma, equalWeight = FALSE) {  
 # Simulation  
 set.seed(123)  
 sim = sim(sigma)  
   
 # Particle Filtering  
 set.seed(123)  
 Z = particle\_filtering(sim, sigma, equalWeight)  
   
 # Visualization  
 z = sim$z  
   
 # t = 1  
 visualize(z, Z, t = 1)  
   
 # t = 50  
 visualize(z, Z, t = 50)  
   
 # t = 99  
 visualize(z, Z, t = 99)  
   
 # t = 100  
 visualize(z, Z, t = 100)  
}

## Task 1

In this task we are to implement the given SSM and simulate for T = 100 time steps. Using M = 100 particles and the 100 observations we then use particle filtering to try and identify the state.

We can see that the particles are very uncertain in the first time step (reasonable given its uniform) whereas it is more certain for future time steps. The expected location is never on point with the true location however it is rather close with a rather small standard distrubtion on the particles.

sigma = 1  
execute(sigma)



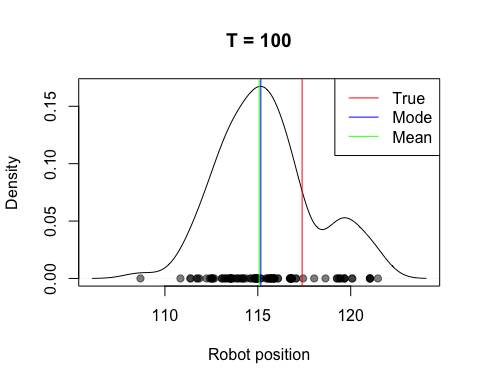
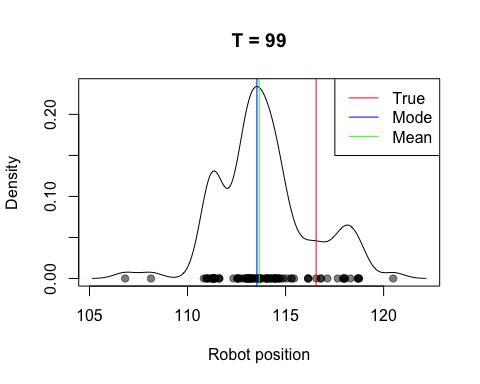
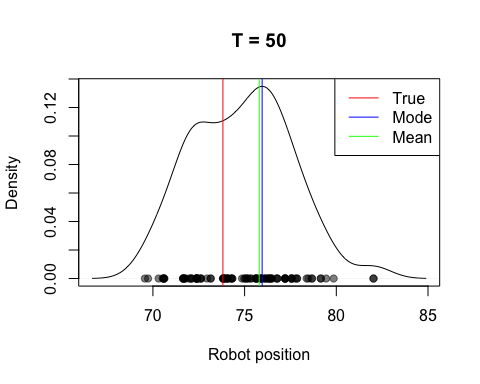
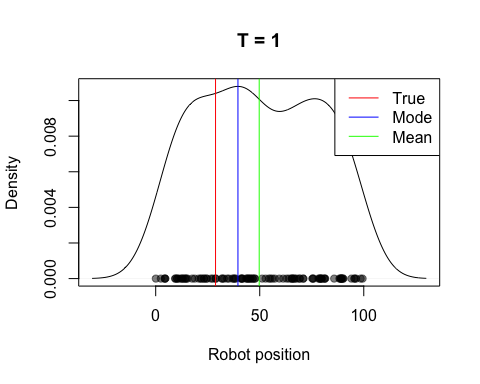
## Task 2

Repeat previous task with emission standard deviation of 5 & 50

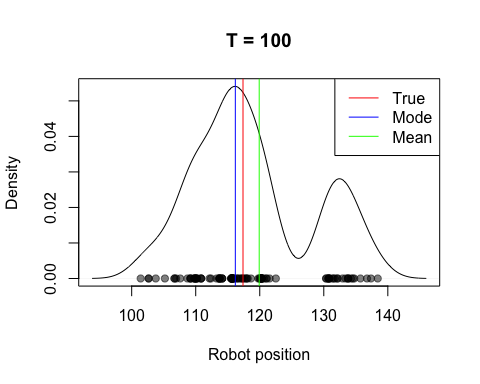
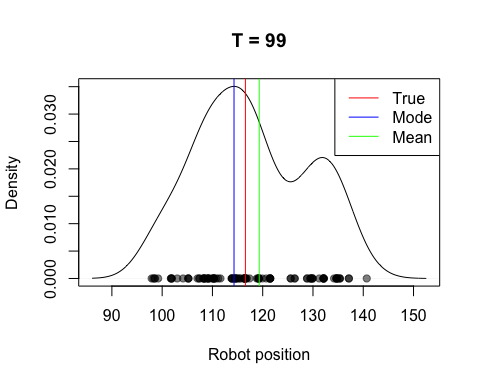
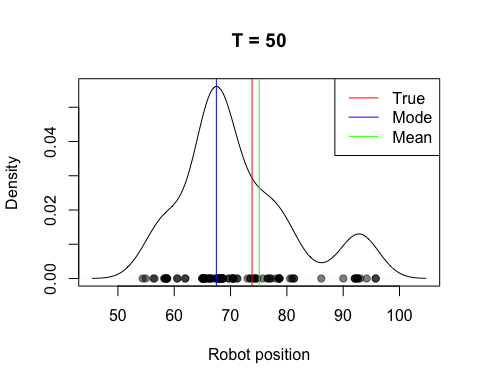
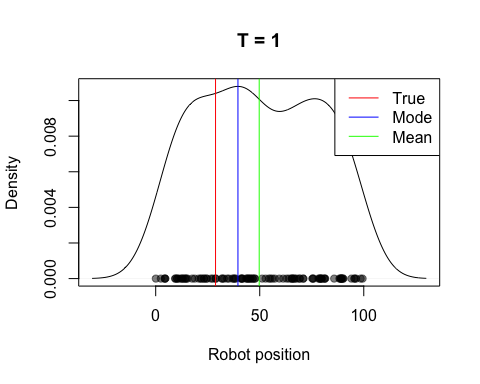
We can see that even though the distrubution of the particles is a lot wider the expected location is still close, regardless of emission standard deviation. It performs worse than previous task, however it still follows the trend of the true value and gives reasonable expected locations.

This is because the emission deviation is accounted for in the weight calculates as well as being symmetric which results in the more “spread” knowledge still having its mass around the true location.

# Sigma = 5  
sigma = 5  
execute(sigma)



# Sigma = 50  
sigma = 50  
execute(sigma)

 ## Task 3 Particle filtering with equal weights

Here the result is way of for each time step. We can see that the sampled particles doesn’t even come close to the true location.

This happens because if the weights are equal for each particle the correction sampling does not take into account more probable particles. Thus, the distrubtion of the final particles can be leaning towards any direction and not follow the more probable “path”.

sigma = 1  
execute(sigma, equalWeight = TRUE)

