SSM_Lab3_GroupReport

Jasleen(jasma846),Maria(marse306),Tejashree(tejma768),Vinay(vinbe289)

October 7, 2019

The purpose of the lab is to put in practice some of the concepts covered in the lectures. To do so, you are asked to implement the particle filter for robot localization. The robot moves along the horizontal axis according to the following SSM:

Transition model : $p(z_t|z_{t-1}) = (\mathcal{N}(z_t|z_{t-1},1) + \mathcal{N}(z_t|z_{t-1}+1,1) + \mathcal{N}(z_t|z_{t-1}+2,1))/3$

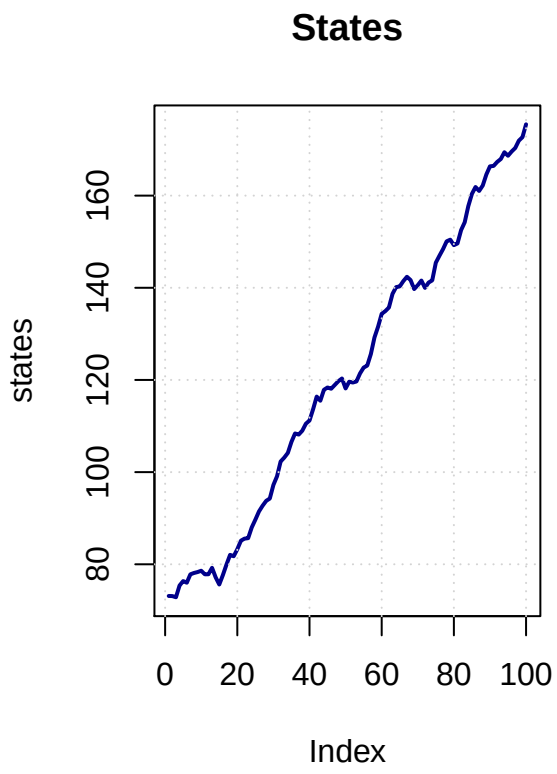
Emission Model : $p(x_t|z_t) = (\mathcal{N}(x_t|z_t,1) + \mathcal{N}(x_t|z_t-1,1) + \mathcal{N}(x_t|z_t+1,1))/3$

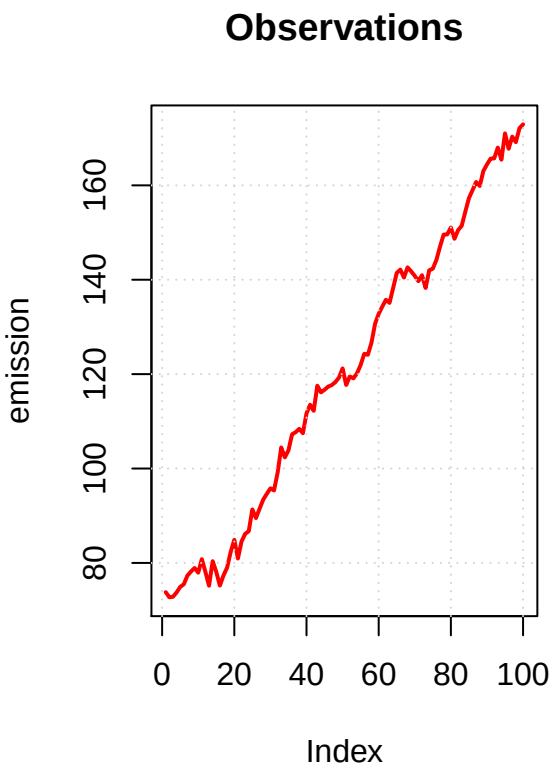
Initial MOdel : $p(z_1) = Uniform(0,100)$

Question 1

Implement the SSM above. SImulate it for T = 100 time steps to obtain $z_{1:100)}$ (i.e states) and $x_{1:100}$ (i.e observations) .Use the observations (i.e., sensor readings) to identify the state (i.e., robot location) via particle filtering. Use 100 particles. Show the particles, the expected location and the true location for the first and last time steps, as well as for two intermediate time steps of your choice.

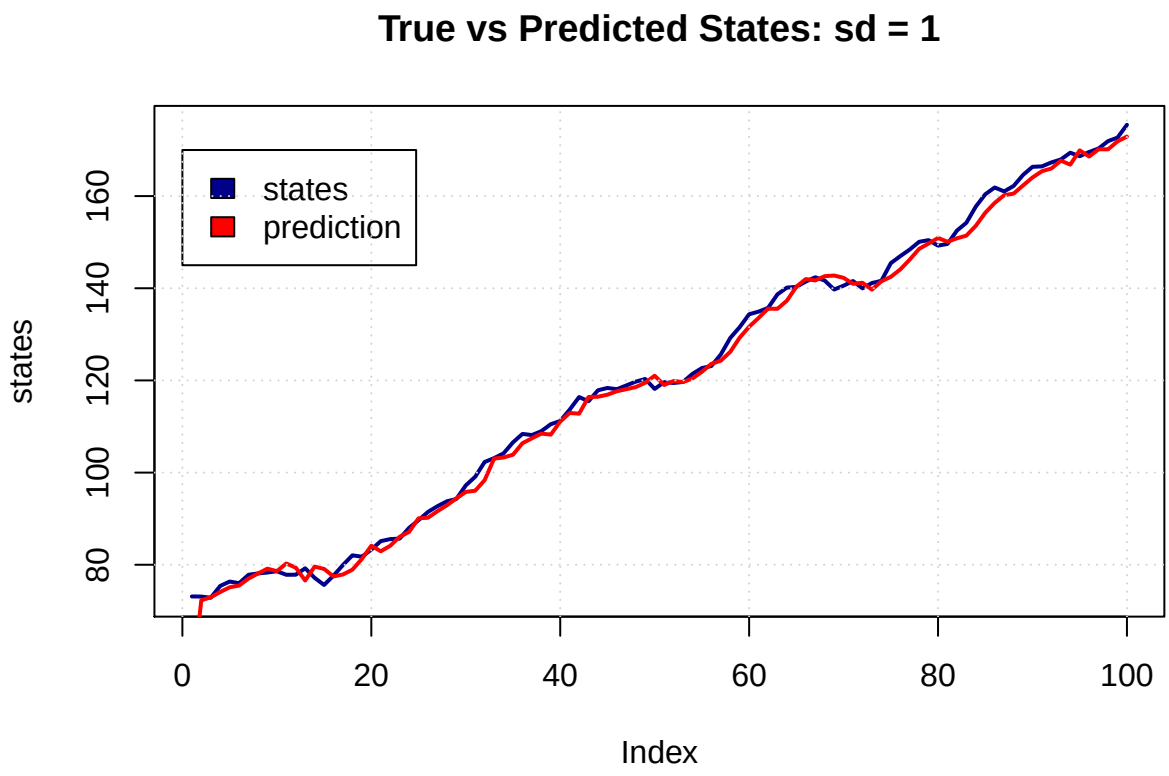
```
#samples
#z = state variable
#x = observed variable
set.seed(12345)
generateSamples = function(T , sd_emission){
  z = c()
  x = c()
  z[1] = runif(1,0,100)
  for(i in 1:T){
    c1 = sample(c(1,2,3), size = 1)
    mux = ifelse(c1 == 1,z[i],ifelse(c1==2,z[i]-1,z[i]+1))
    x[i] = rnorm(1, mean = mux, sd = sd_emission)
    c2 = sample(c(1,2,3), size = 1)
    muz = ifelse(c1 == 1,z[i],ifelse(c1==2,z[i]+1,z[i]+2))
    z[i+1] = rnorm(1, mean = muz, sd = 1)
  }
  return(data.frame("states" = z[2:101] , "observations" = x[1:100]))
}

T = 100
samples = generateSamples(T , 1)
states = samples$states
emission = samples$observations
```

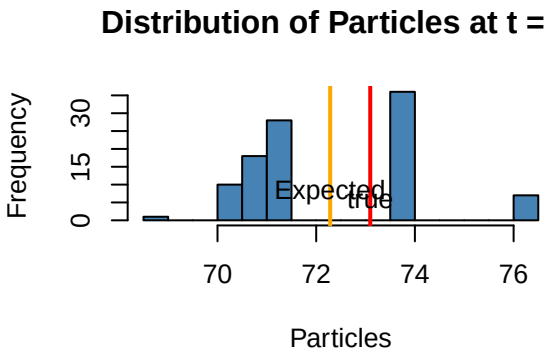


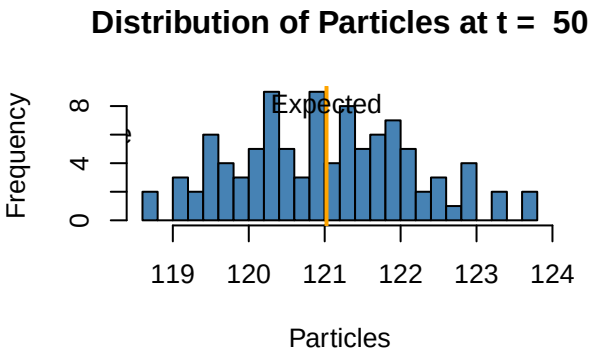


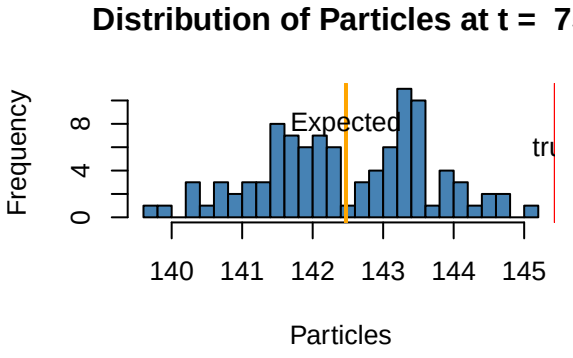

```
#Particle filter
particleFilter = function(sd_emission, M, T , emission, weights = TRUE){
  z =  runif(M,0,100) # Initial particles from uniform distribution
  bel_bar = c()
  bel = matrix(NA , M, T)
  bel[,1] = z
  wt = c()

  for(t in 2:T){
    for(m in 1:M){
        c1 = sample(c(1,2,3), size = 1)
        muz = ifelse(c1 == 1,z[m],ifelse(c1==2,z[m]+1,z[m]+2))

        # sampling m'th particle from transmission model
        bel_bar[m] = rnorm(1, mean = muz, sd = 1)
        if(weights){
           wt[m] = (dnorm(emission[t],bel_bar[m],sd_emission)+
                    dnorm(emission[t],bel_bar[m]-1,sd_emission)+
                    dnorm(emission[t],bel_bar[m]+1,sd_emission))/3
        }else{
          wt[m] = 1/M

        }
    }
    #wt = wt/sum(wt) # normalization
    z = sample(bel_bar, M, replace = TRUE, prob = wt) # resampling - weighted sampling
```

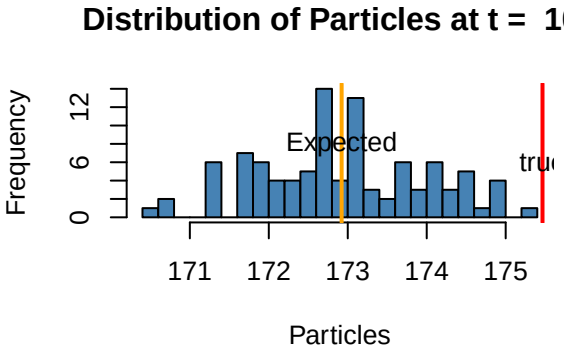
```
    bel[,t] = z
  }
  return(bel)
}
```





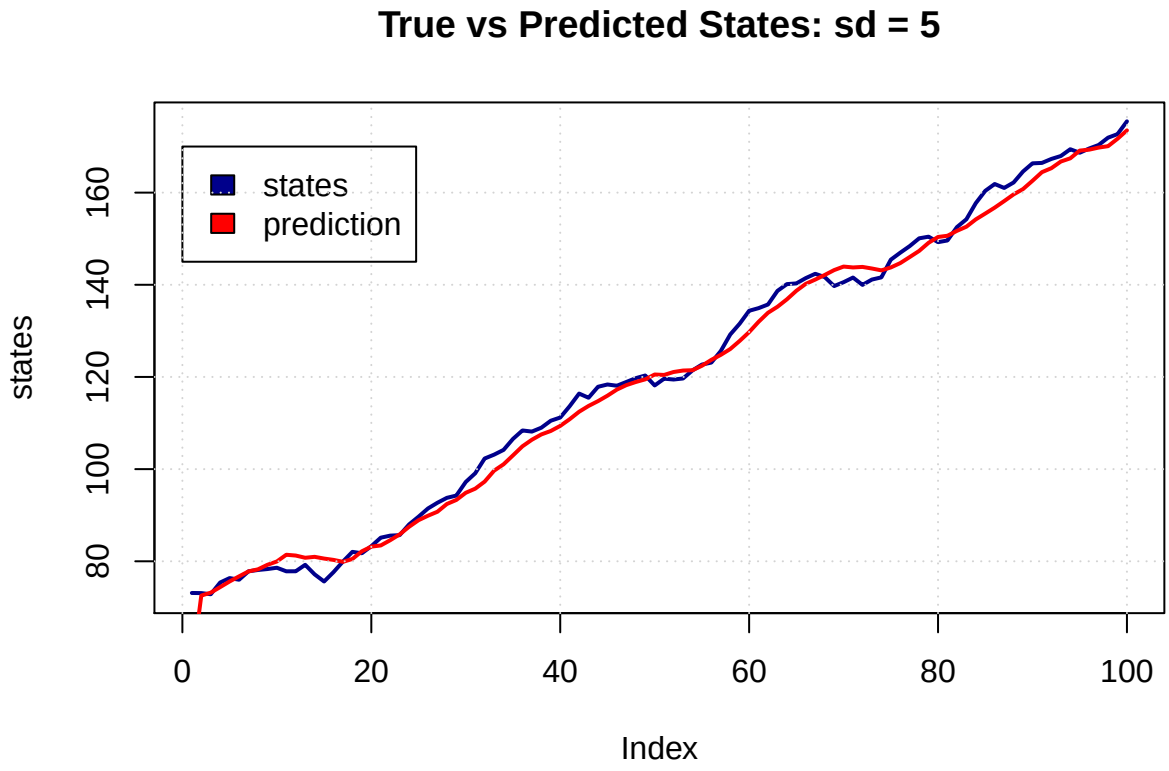


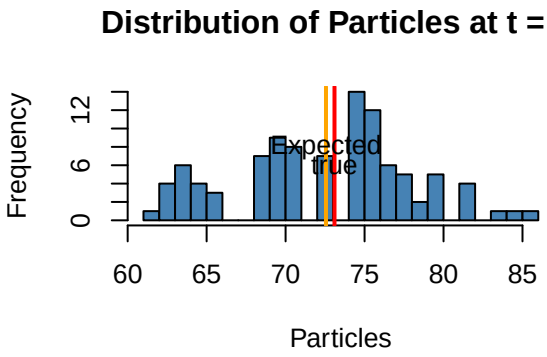


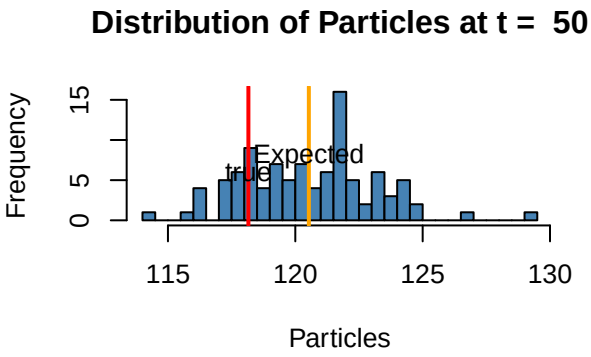


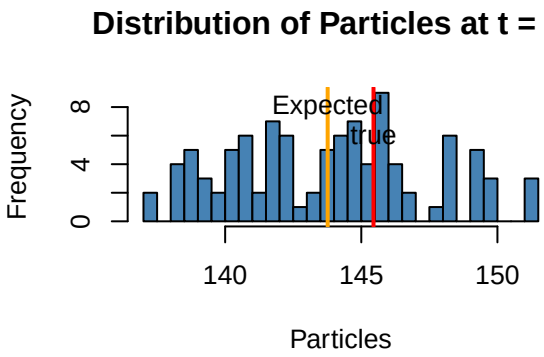
Question 2

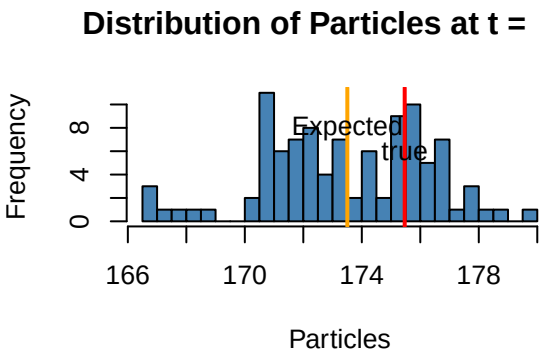
Repeat the excercise above replacing the standard deviation of the emission model with 5 and 50. Comment on how this affects the results

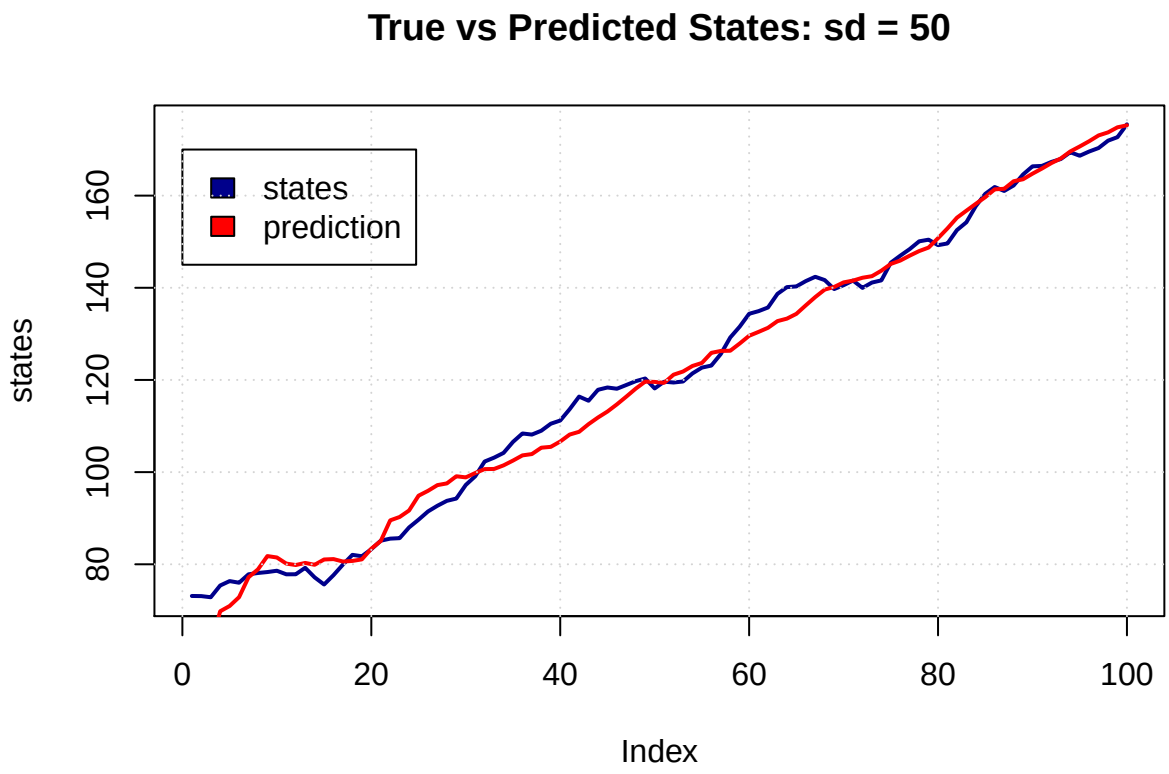

True vs Predicted States: sd = 5
states
prediction
states
80
100
120
140
160
0
20
40
60
80
100
Index

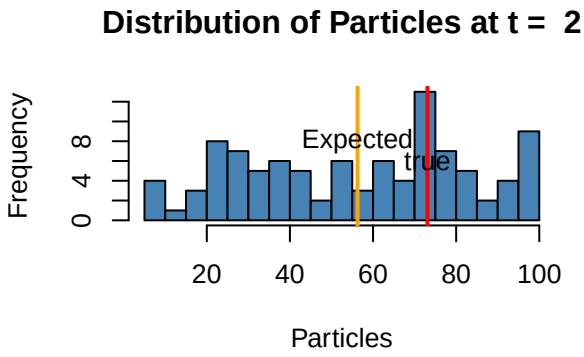

Distribution of Particles at t = 2
Frequency
0
6
12
Expected
true
60
65
70
75
80
85
Particles

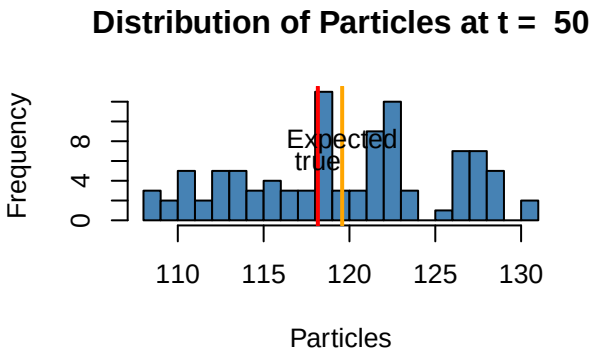

Distribution of Particles at t = 50
Frequency
0
5
15
true
Expected
115
120
125
130
Particles

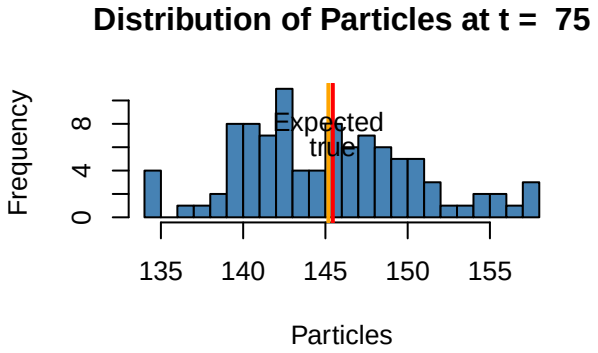

Distribution of Particles at t = 75
Frequency
0
4
8
Expected
true
140
145
150
Particles

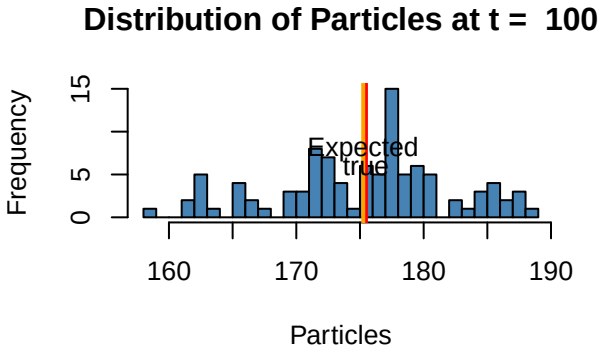

Distribution of Particles at t = 100
Frequency
0
4
8
Expected
true
166
170
174
178
Particles


True vs Predicted States: sd = 50
states
prediction
states
80
100
120
140
160
0
20
40
60
80
100
Index





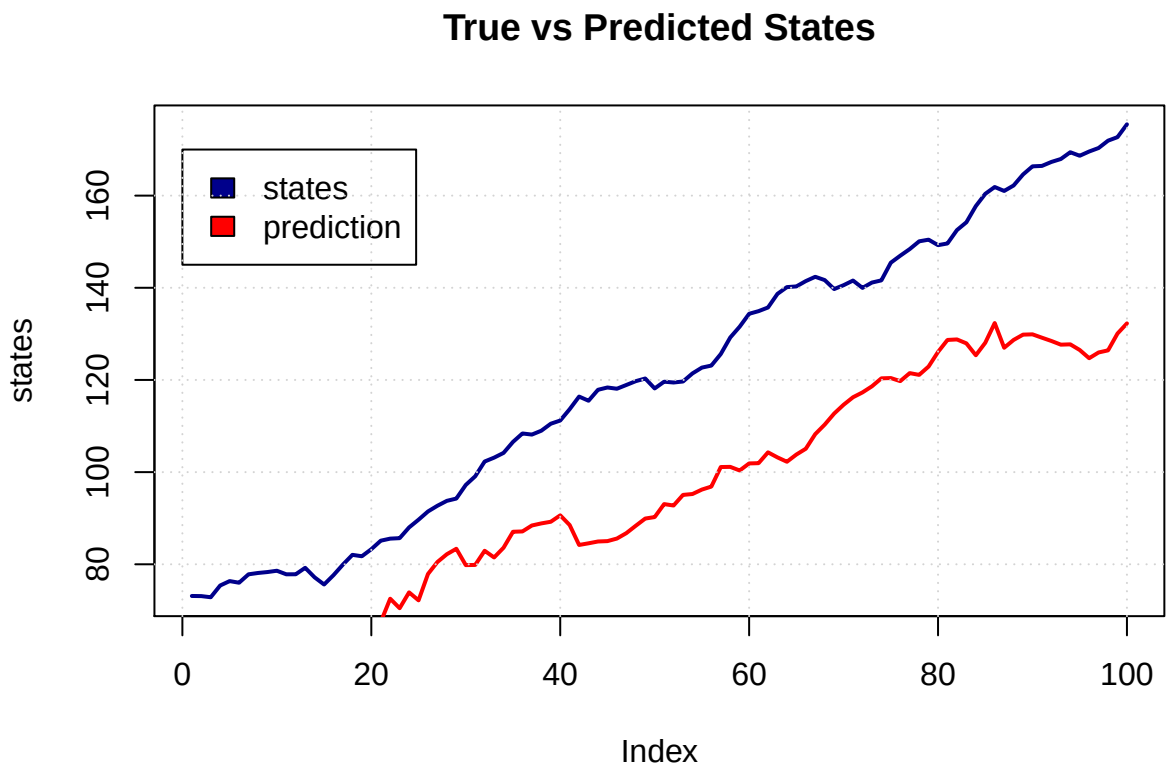


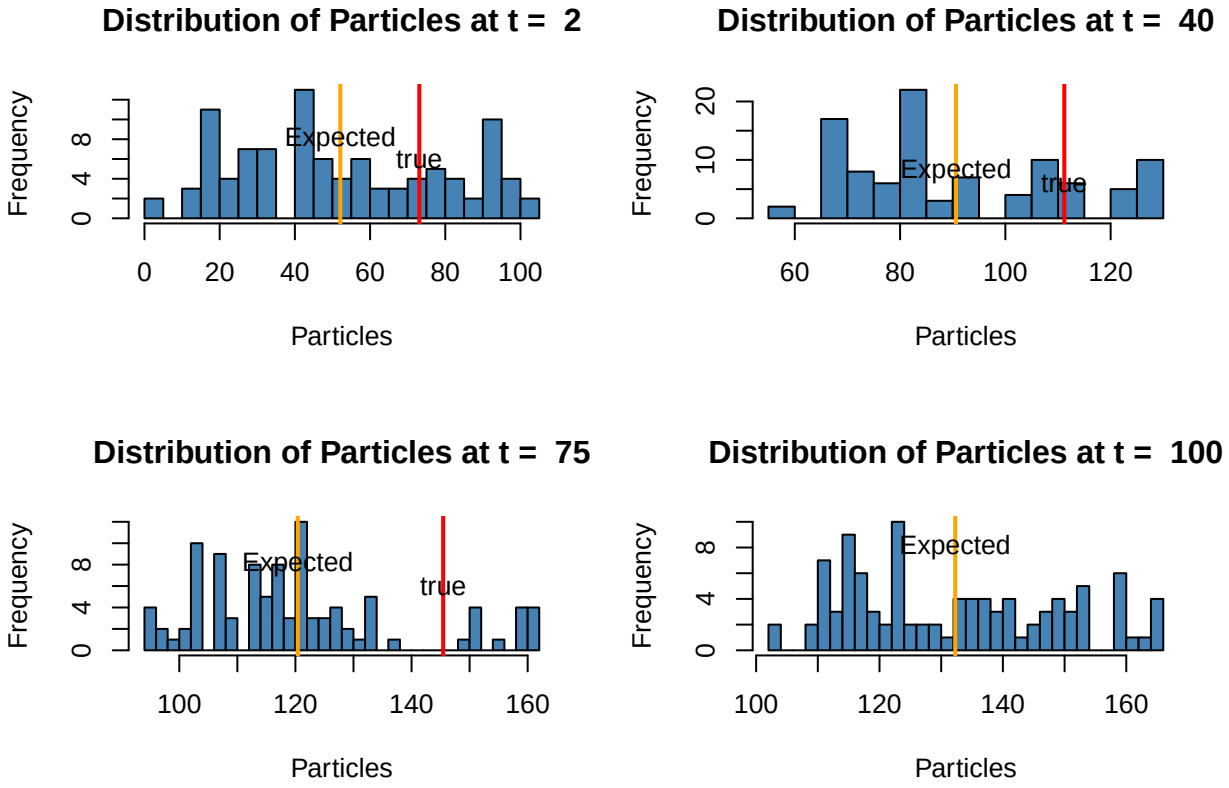


As we can see in the plot, the prediction is getting less accurate as standard deviation increases. In model, as we are introducing high standard deviation, variance and uncertainty in model is more which reflects in our prediction.

Question 3

Finally show and explain what happens when weights in the particle filter are always equal to 1 i.e there is no correction.


True vs Predicted States
states
prediction
states
80
100
120
140
160
0
20
40
60
80
100
Index



In particle filter, we are trying to approximate the posterior as a set of random samples drawn from posterior. If we do not change the weights, importance sampling is not happening and hence no correction. So the set of particles will be always random and it will never correct itself to the true density. So prediction is expected not to be accurate.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
#samples
#z = state variable
#x = observed variable
set.seed(12345)
generateSamples = function(T , sd_emission){
  z = c()
  x = c()
  z[1] = runif(1,0,100)
  for(i in 1:T){
    c1 = sample(c(1,2,3), size = 1)
    mux = ifelse(c1 == 1,z[i],ifelse(c1==2,z[i]-1,z[i]+1))
    x[i] = rnorm(1, mean = mux, sd = sd_emission)
    c2 = sample(c(1,2,3), size = 1)
    muz = ifelse(c1 == 1,z[i],ifelse(c1==2,z[i]+1,z[i]+2))
    z[i+1] = rnorm(1, mean = muz, sd = 1)
  }
  return(data.frame("states" = z[2:101] , "observations" = x[1:100]))
```

```
}

T = 100
samples = generateSamples(T , 1)
states = samples$states
emission = samples$observations
par(mfrow =c(1,2))
plot(states , type = "l" , lwd = 2 , col = "darkblue" , main = "States")
grid()
plot(emission , type = "l" , col = "red" , lwd = 2 , main = "Observations")
grid()
#Particle filter
particleFilter = function(sd_emission, M, T , emission, weights = TRUE){
  z =  runif(M,0,100) # Initial particles from uniform distribution
  bel_bar = c()
  bel = matrix(NA , M, T)
  bel[,1] = z
  wt = c()

  for(t in 2:T){
    for(m in 1:M){
        c1 = sample(c(1,2,3), size = 1)
        muz = ifelse(c1 == 1,z[m],ifelse(c1==2,z[m]+1,z[m]+2))

        # sampling m'th particle from transmission model
        bel_bar[m] = rnorm(1, mean = muz, sd = 1)
        if(weights){
           wt[m] = (dnorm(emission[t],bel_bar[m],sd_emission)+
                    dnorm(emission[t],bel_bar[m]-1,sd_emission)+
                    dnorm(emission[t],bel_bar[m]+1,sd_emission))/3
        }else{
          wt[m] = 1/M

        }
    }
    #wt = wt/sum(wt) # normalization
    z = sample(bel_bar, M, replace = TRUE, prob = wt) # resampling - weighted sampling
    bel[,t] = z
  }
  return(bel)
}

particles = particleFilter(sd_emission = 1, M=100 , T=100 , emission = emission)
plot(states , type = "l" , lwd = 2 , col = "darkblue",
     main = "True vs Predicted States: sd = 1")
lines(colMeans(particles),col = "red",lwd = "2")
legend(0,170,c("states","prediction"),c("darkblue","red"))
grid()
#Visualization
#posterior distribution of particles at time t = 100
visualizeParticle = function(t, states , particles){
  hist(particles[,t], breaks = 25,
  xlab="Particles", col="steelblue",border="black",
```

```
  main=paste("Distribution of Particles at t = ",t))
  abline(v = states[t], col = 'red' , lwd = 2)
  abline(v = mean(particles[,t]) , col = "orange" , lwd = 2)
  text(mean(particles[,t]), 8, "Expected", col = "black")
  text(states[t], 6, "true", col = "black")
}

par(mfrow = c(2,2))
visualizeParticle(2,states = states , particles = particles)
visualizeParticle(50,states = states , particles = particles)
visualizeParticle(75,states = states , particles = particles)
visualizeParticle(100,states = states , particles = particles)


samples_5 = generateSamples(T , 5)
states_5 = samples$states
emission_5 = samples$observations


particles_5 = particleFilter(sd_emission = 5, M=100 , T=100 , emission = emission_5)
plot(states , type = "l" , lwd = 2 , col = "darkblue",
     main = "True vs Predicted States: sd = 5")
lines(colMeans(particles_5),col = "red",lwd = "2")
legend(0,170,c("states","prediction"),c("darkblue","red"))
grid()

par(mfrow = c(2,2))
visualizeParticle(2,states = states_5 , particles = particles_5)
visualizeParticle(50,states = states_5 , particles = particles_5)
visualizeParticle(75,states = states_5 , particles = particles_5)
visualizeParticle(100,states = states_5 , particles = particles_5)

samples_50 = generateSamples(T , 50)
states_50 = samples$states
emission_50 = samples$observations


particles_50 = particleFilter(sd_emission = 50, M=100 , T=100 , emission = emission_50)
plot(states , type = "l" , lwd = 2 , col = "darkblue",
     main = "True vs Predicted States: sd = 50")
lines(colMeans(particles_50),col = "red",lwd = "2")
legend(0,170,c("states","prediction"),c("darkblue","red"))
grid()


par(mfrow = c(2,2))
visualizeParticle(2,states = states_50 , particles = particles_50)
visualizeParticle(50,states = states_50 , particles = particles_50)
visualizeParticle(75,states = states_50 , particles = particles_50)
visualizeParticle(100,states = states_50 , particles = particles_50)

particles = particleFilter(sd_emission=1, M=100 , T=100 , emission = emission, weights = FALSE)
plot(states , type = "l" , lwd = 2 , col = "darkblue",
```

```
     main = "True vs Predicted States")
lines(colMeans(particles),col = "red",lwd = "2")
legend(0,170,c("states","prediction"),c("darkblue","red"))
grid()


par(mfrow = c(2,2))
visualizeParticle(2,states = states , particles = particles)
visualizeParticle(40,states = states , particles = particles)
visualizeParticle(75,states = states , particles = particles)
visualizeParticle(100,states = states , particles = particles)
```