AML Lab 3 (State Space Models)

Roshni Sundaramurthy

30/09/2019

Contents

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. For the particle filter algorithm, please check Section 13.3.4 of Bishop's book and/or the slides for the last lecture on state space models (SSMs). The robot moves along the horizontal axis according to the following SSM:

Transition model:

$$p(z_t|z_{t-1}) = (N(z_t|z_{t-1}, 1) + N(z_t|z_{t-1} + 1, 1) + N(z_t|z_{t-1} + 2, 1))/3$$

Emission model:

$$p(x_t|z_t) = (N(x_t|z_t, 1) + N(x_t|z_t - 1, 1) + N(x_t|z_t + 1, 1))/3$$

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.

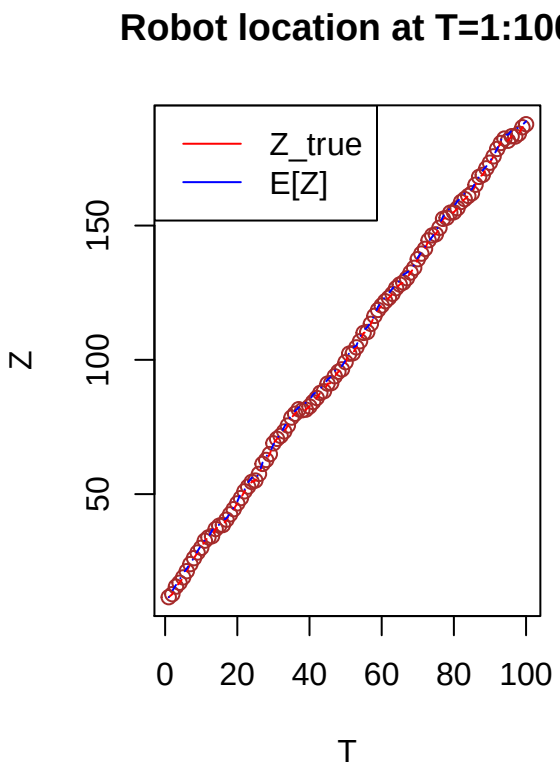
```
# transition model
transition = function(z,stdn,pos){
  if(pos==1/3){rnorm(1,mean=z+0,sd=stdn)}
  ifelse(pos==2/3,rnorm(1,mean=z+1,sd=stdn),rnorm(1,mean=z+2,sd=stdn))
}

# emission model
emission = function(x,stdn,pos){
  if(pos==1/3){rnorm(1,mean=x+0,sd=stdn)}
  ifelse(pos==2/3,rnorm(1,mean=x-1,sd=stdn),rnorm(1,mean=x+1,sd=stdn))
}

# Initial model
initial<-function(n){
  return (runif(n,0,100))
}

# density emission
den_emi_dis<-function(x,z,stdn){
  return ((dnorm(x,mean=z,sd=stdn)+dnorm(x,mean=z-1,sd=stdn)+dnorm(x,mean=z+1,sd=stdn))/3)
}

###################### STATE SPACE MODEL  ########################################
z_true=0
x_obs=0

simSSM = function(T,stdn_z, stdn_x){

  for(t in seq(1,T)){
    # t==1 -> initial state
    z_true[t] = ifelse(t==1,initial(100),transition(z_true[t-1], stdn_z,
                                                    pos = runif(100,0,1)[t]))

    x_obs[t] = emission(z_true[t], stdn_x, pos = runif(100,0,1)[t])
  }

  return(list(z_true,x_obs))

}
```

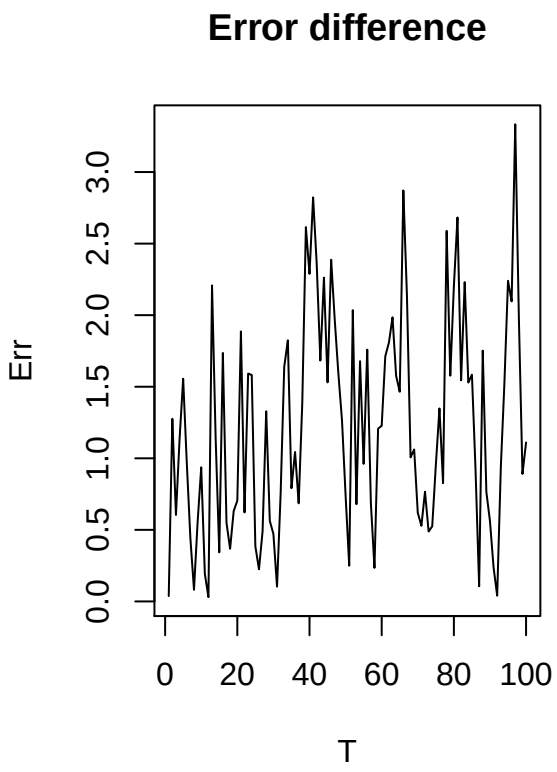
```
data=simSSM(100,1,1)
z_true=unlist(data[1]) # True location
```

```
x_obs = unlist(data[2]) # Observations
```

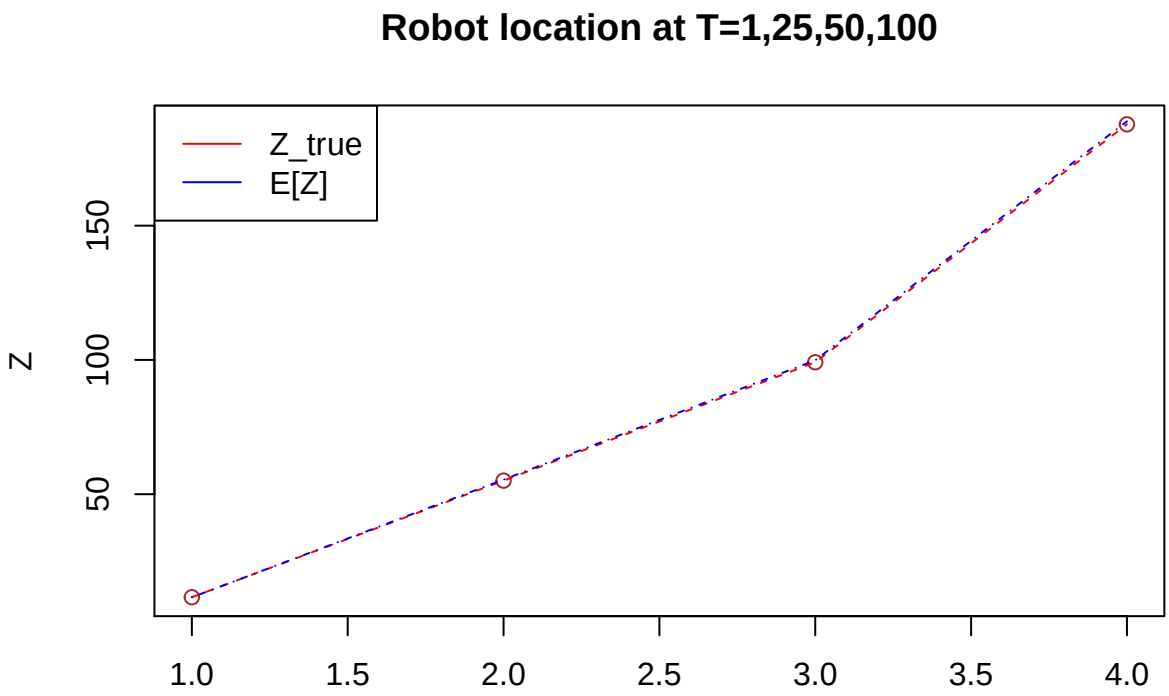
Standard deviation of the emission model with 1

```
filter = ParticleFilter1(x_obs,stdn_x=1,stdn_z=1)
```





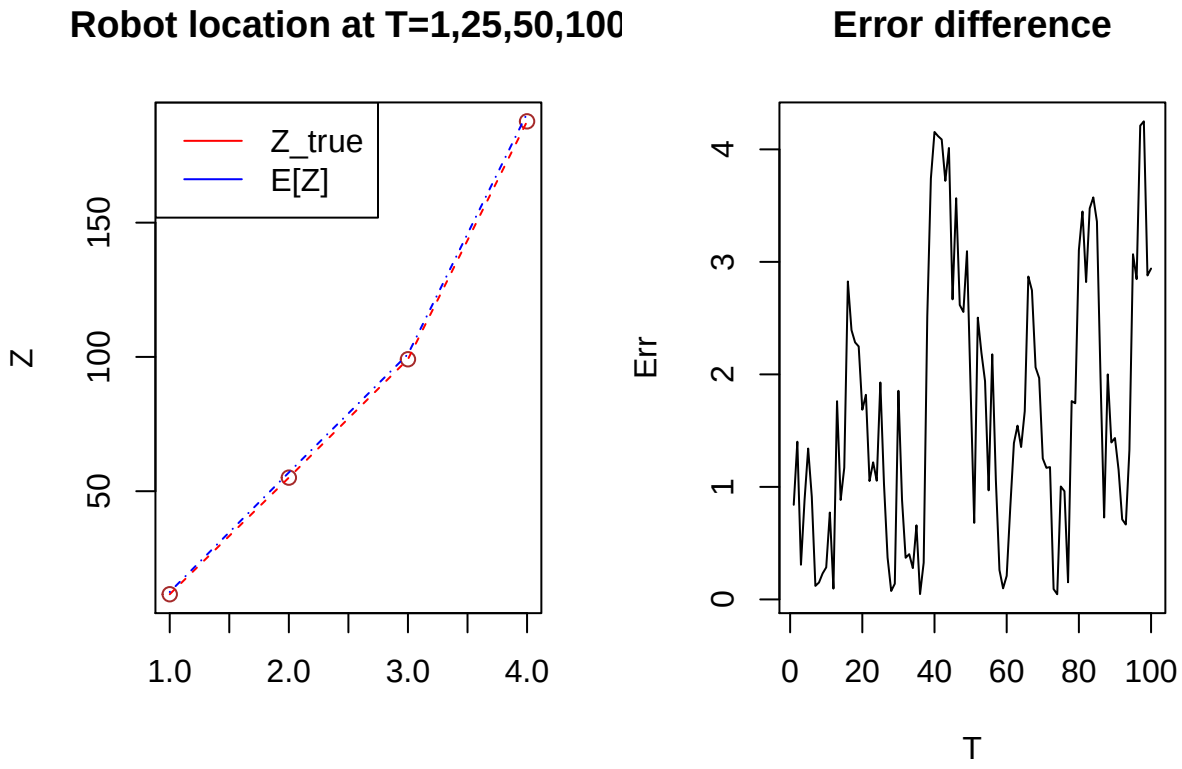

```
# for T=1,25,50,100
plot_it(filter[[1]],filter[[2]])
```



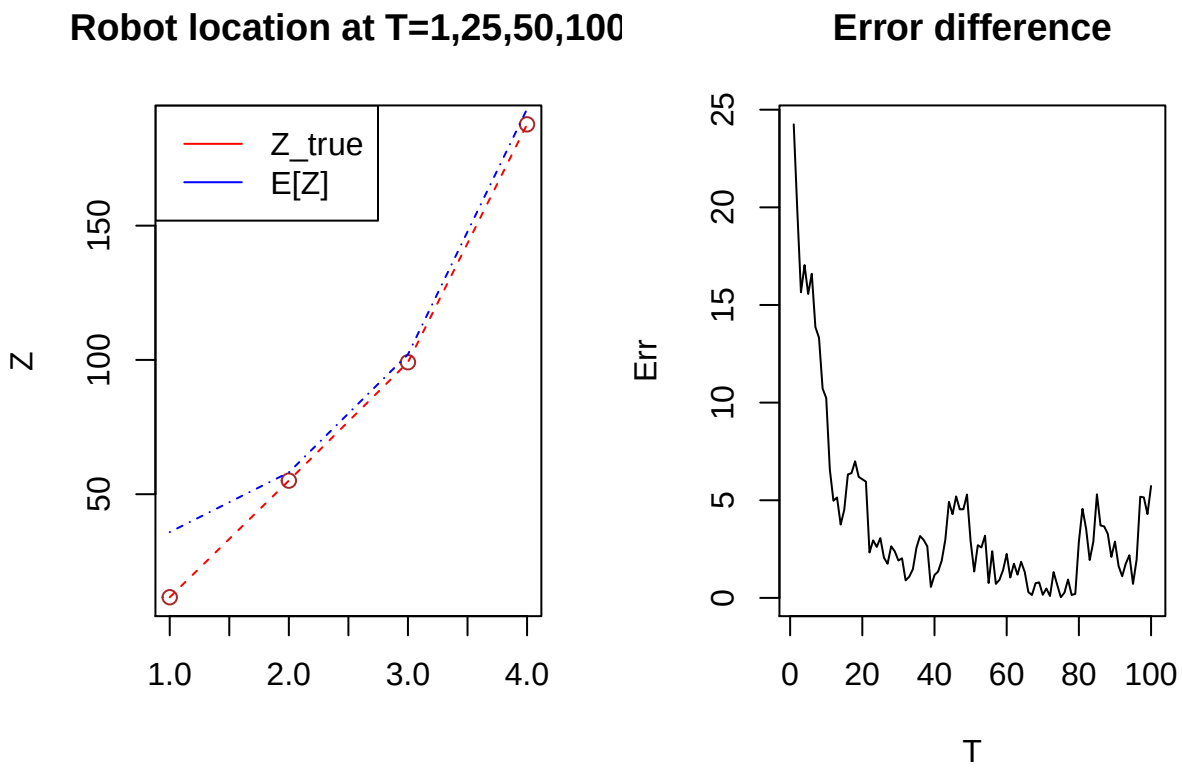
From the plot, we can say that the true (red) and expected (blue) values are almost closer to each other.

Question 2: Repeat the exercise above replacing the standard deviation of the emission model with 5 and then with 50. Comment on how this affects the results.

```
par(mfrow=c(1,2))
# Standard deviation of the emission model with 5
filter5 = ParticleFilter1(x_obs,stdn_x = 5, stdn_z=1)
plot_it(filter5[[1]],filter5[[2]])
# Error difference
plot(filter5[[3]], type="l", ylab = "Err", xlab = "T",
     main="Error difference")
```



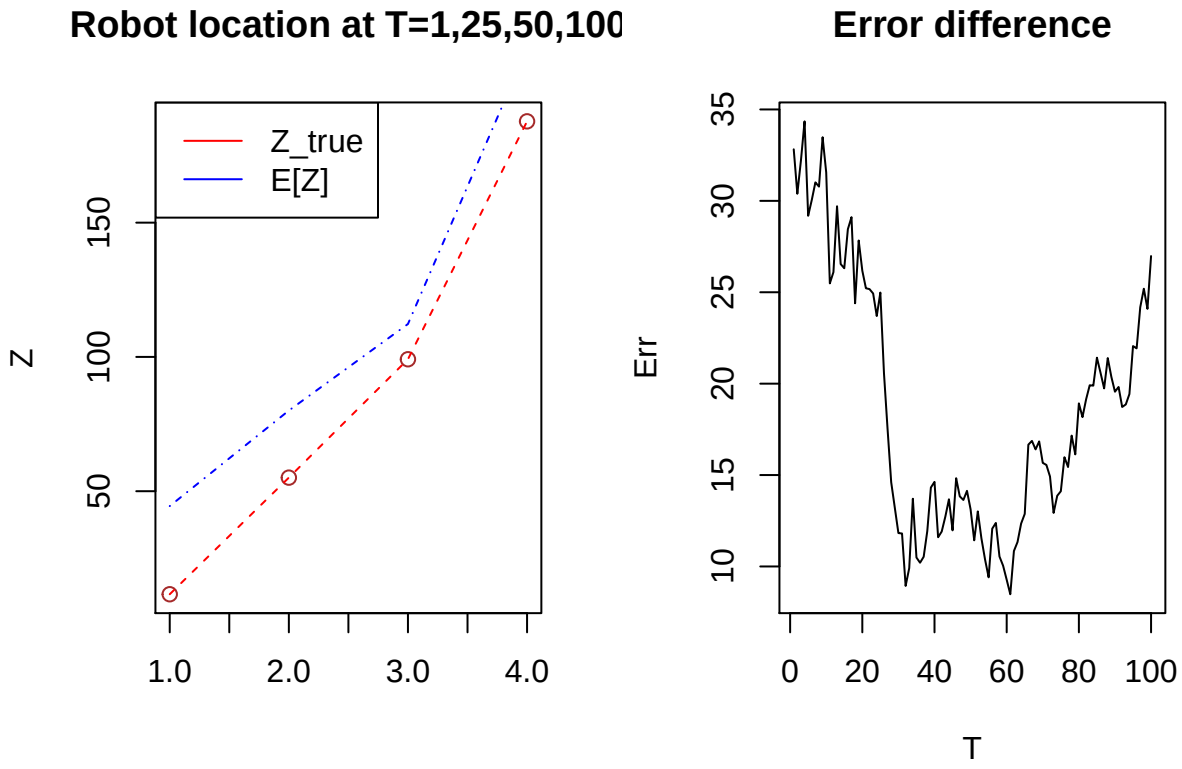

```
par(mfrow=c(1,2))
# Standard deviation of the emission model with 50
filter50 = ParticleFilter1(x_obs,stdn_x = 50,stdn_z=1)
plot_it(filter50[[1]],filter50[[2]])
# Error difference
plot(filter50[[3]], type="l", ylab = "Err", xlab = "T",
     main="Error difference")
```



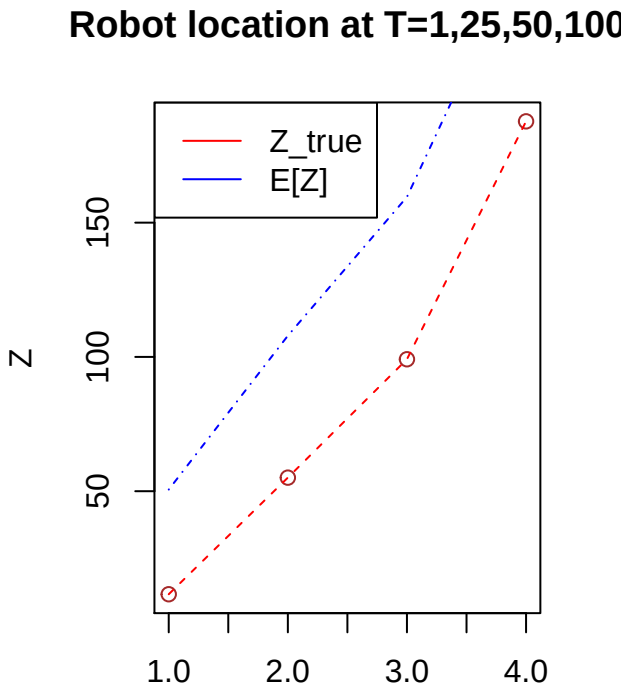
On increasing the standard deviation values of emission model, the uncertainty of robot localization increases. The true (red) and expected (blue) value differs.

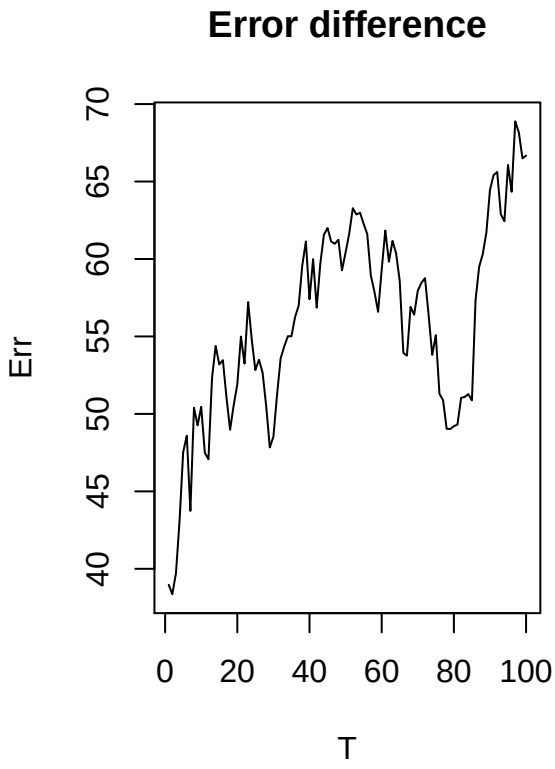
Question 3: Finally, show and explain what happens when the weights in the particle filter are always equal to 1, i.e. there is no correction.

```
par(mfrow=c(1,2))
# Standard deviation of the emission model with 1
filter1 = ParticleFilter2(x_obs,stdn_x=1,stdn_z=1)
plot_it(filter1[[1]],filter1[[2]])
# Error difference
plot(filter1[[3]], type="l", ylab = "Err", xlab = "T",
     main="Error difference")
```

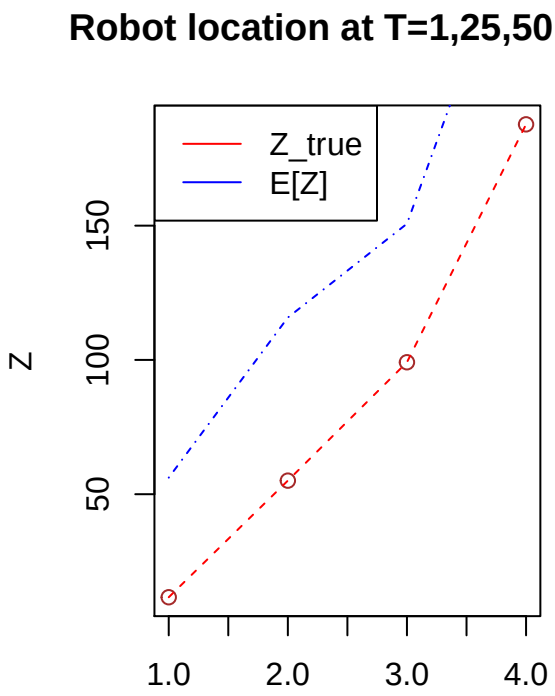


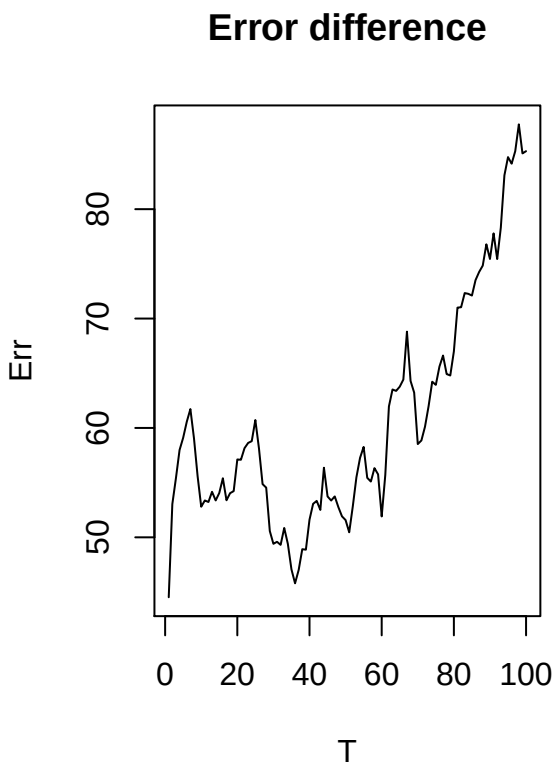

```
# Standard deviation of the emission model with 5
filter5 = ParticleFilter2(x_obs,stdn_x=1,stdn_z=1)
plot_it(filter5[[1]],filter5[[2]])
# Error difference
plot(filter5[[3]], type="l", ylab = "Err", xlab = "T",
     main="Error difference")
```






```
# Standard deviation of the emission model with 50
filter50 = ParticleFilter2(x_obs,stdn_x=1,stdn_z=1)
plot_it(filter50[[1]],filter50[[2]])
# Error difference
plot(filter50[[3]], type="l", ylab = "Err", xlab = "T",
     main="Error difference")
```





When the weights in the particle filter are not updated, it is of no use because for every time steps, we reset the weight value to 1. So, the probability of drawing every particles is disturbed. Hence, it is evident from the plot that the uncertainty of robot localization increases greatly and prediction is very poor.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)

# transition model
transition = function(z,stdn,pos){
  if(pos==1/3){rnorm(1,mean=z+0,sd=stdn)}
  ifelse(pos==2/3,rnorm(1,mean=z+1,sd=stdn),rnorm(1,mean=z+2,sd=stdn))
}

# emission model
emission = function(x,stdn,pos){
  if(pos==1/3){rnorm(1,mean=x+0,sd=stdn)}
  ifelse(pos==2/3,rnorm(1,mean=x-1,sd=stdn),rnorm(1,mean=x+1,sd=stdn))
}

# Initial model
initial<-function(n){
  return (runif(n,0,100))
}
```

```r
# density emission
den_emi_dis<-function(x,z,stdn){
  return ((dnorm(x,mean=z,sd=stdn)+dnorm(x,mean=z-1,sd=stdn)+dnorm(x,mean=z+1,sd=stdn))/3)
}

###################### STATE SPACE MODEL  ########################################
z_true=0
x_obs=0

simSSM = function(T,stdn_z, stdn_x){

  for(t in seq(1,T)){
    # t==1 -> initial state
    z_true[t] = ifelse(t==1,initial(100),transition(z_true[t-1], stdn_z,
                                                    pos = runif(100,0,1)[t]))

    x_obs[t] = emission(z_true[t], stdn_x, pos = runif(100,0,1)[t])
  }

  return(list(z_true,x_obs))

}

data=simSSM(100,1,1)
z_true=unlist(data[1]) # True location
x_obs = unlist(data[2]) # Observations
################### PARTICLE FILTER ###################
# Initialize

T = 100 # time steps
n_par = 100 # Use 100 particles
bel<-matrix(0,nrow = 100, ncol = T + 1) # n_par=100
w<-matrix(0,nrow = 100,ncol = T) # weights
com<-matrix(0,100,T) # component
Ezt <- vector(length=T) # expected location
err<-vector(length=T) # error = z_true - E(z)

# function for filtering

ParticleFilter1 = function(x_obs,stdn_x,stdn_z){

  bel[,1] <- initial(100)
  for(t in seq(1,T)){
    for(i in seq(1,n_par)){
      # calculate weights (probability of evidence given sample from z)
      w[i,t]<- den_emi_dis(x_obs[t],bel[i,t],stdn_x)
    }
    # calculate and normalize weights
    w[,t]<-w[,t]/sum(w[,t])

    # weighted resampling with replacement
    com[,t]<-sample(bel[,t],n_par,replace=TRUE,prob=w[,t])
    for (i in seq(1,n_par)){
```

```
      bel[i, t+1] <- transition(z=com[i, t], stdn = stdn_z, pos = runif(100,0,1)[i])
    }

    Ezt[t]<-sum(w[,t] * bel[,t])
    err[t]<-abs(z_true[t]-Ezt[t])

    #cat("t: ",t,", z_t: ",z_true[t],", E[z_t]: ",Ezt[t]," , error: ",err[t],"\n")
    #flush.console()

  }
  return(list(z_true, Ezt, err, bel))
}

filter = ParticleFilter1(x_obs,stdn_x=1,stdn_z=1)
# plot function for T=1,25,50,100

plot_it = function(z,Ezt){
  plot(z[c(1,25,50,100)], col="brown", ylab = "Z",
       xlab="",main = "Robot location at T=1,25,50,100")
  lines(z[c(1,25,50,100)], col="red", type = "l", lty=2)
  lines(Ezt[c(1,25,50,100)], col="blue", type = "l", lty=4)
  legend("topleft", legend = c("Z_true", "E[Z]"),
       col = c("red","blue"), lty=c(1,1))
}

par(mfrow=c(1,2))
# for T=1:100
plot(filter[[1]], col="brown", ylab = "Z", xlab = "T", main = "Robot location at T=1:100")
lines(filter[[1]], col="red", type = "l", lty=2)
lines(filter[[2]], col="blue", type = "l", lty=4)
legend("topleft", legend = c("Z_true", "E[Z]"),
       col = c("red","blue"), lty=c(1,1))

# Error difference
plot(filter[[3]], type="l", ylab = "Err", xlab = "T",
     main="Error difference")

# for T=1,25,50,100
plot_it(filter[[1]],filter[[2]])
par(mfrow=c(1,2))
# Standard deviation of the emission model with 5
filter5 = ParticleFilter1(x_obs,stdn_x = 5, stdn_z=1)
plot_it(filter5[[1]],filter5[[2]])
# Error difference
plot(filter5[[3]], type="l", ylab = "Err", xlab = "T",
     main="Error difference")

par(mfrow=c(1,2))
# Standard deviation of the emission model with 50
filter50 = ParticleFilter1(x_obs,stdn_x = 50,stdn_z=1)
plot_it(filter50[[1]],filter50[[2]])
# Error difference
plot(filter50[[3]], type="l", ylab = "Err", xlab = "T",
```

```
     main="Error difference")
ParticleFilter2 = function(x_obs,stdn_x,stdn_z){

  bel[,1] <- initial(100)
  for(t in seq(1,T)){
    for(i in seq(1,n_par)){
      # calculate weights (probability of evidence given sample from z)
      w[i,t]<- den_emi_dis(x_obs[t],bel[i,t],stdn_x)
    }
    # calculate and normalize weights
    # w[,t]<-w[,t]/sum(w[,t]) # with correction
    w[,t]<- rep(1/100,100) # no correction

    # weighted resampling with replacement
    com[,t]<-sample(bel[,t],n_par,replace=TRUE,prob=w[,t])
    for (i in seq(1,n_par)){
      bel[i, t+1] <- transition(z=com[i, t], stdn = stdn_z, pos = runif(100,0,1)[i])
    }

    Ezt[t]<-sum(w[,t] * bel[,t])
    err[t]<-abs(z_true[t]-Ezt[t])

  }
  return(list(z_true, Ezt, err, bel))
}
par(mfrow=c(1,2))
# Standard deviation of the emission model with 1
filter1 = ParticleFilter2(x_obs,stdn_x=1,stdn_z=1)
plot_it(filter1[[1]],filter1[[2]])
# Error difference
plot(filter1[[3]], type="l", ylab = "Err", xlab = "T",
     main="Error difference")

# Standard deviation of the emission model with 5
filter5 = ParticleFilter2(x_obs,stdn_x=1,stdn_z=1)
plot_it(filter5[[1]],filter5[[2]])
# Error difference
plot(filter5[[3]], type="l", ylab = "Err", xlab = "T",
     main="Error difference")

# Standard deviation of the emission model with 50
filter50 = ParticleFilter2(x_obs,stdn_x=1,stdn_z=1)
plot_it(filter50[[1]],filter50[[2]])
# Error difference
plot(filter50[[3]], type="l", ylab = "Err", xlab = "T",
     main="Error difference")
```

Reference

1. Lecture slide (SSM_L2)
2. https://rpubs.com/awellis/180442