Advance Machine Learning Lab 3

Omkar Bhutra (omkbh878)

7 October 2019

Q1. 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 Bishops 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: $p(x_t|z_t) = (N(x_t|z_t, 1) + N(x_t|z_t - 1, 1) + N(x_t|z_t + 1, 1))/3$

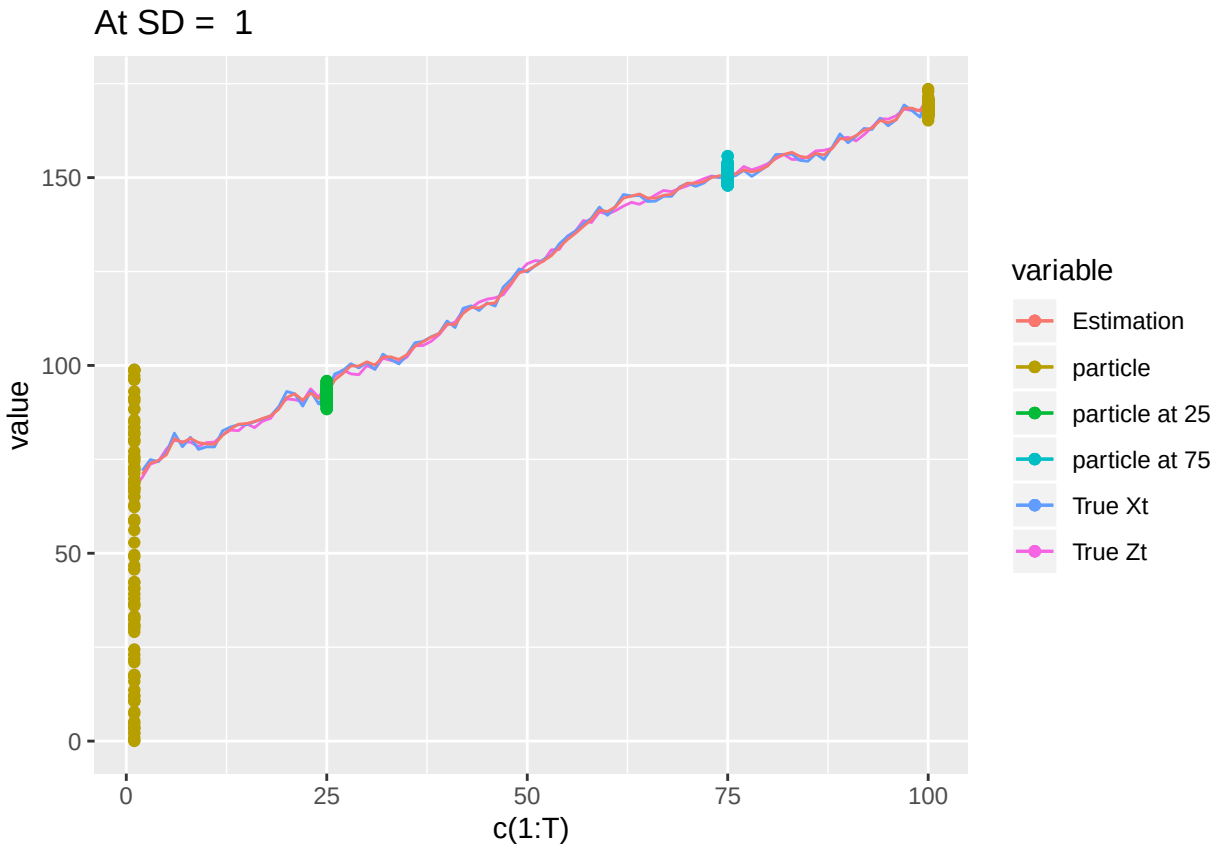
Initial Model: $p(z_1) = Uniform(0, 100)$

A) 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.

```
initModel <- function(len){
  x <- runif(len,0,100)
  return(x)
}

transitionmodel <- function(zt){
  probs = rep(1/3,3)
  draw = sample(1:3,1,prob = probs)
  if(draw==1){
    normTrans <- rnorm(1,zt,transition_sd)
  }
  else if(draw==2){
    normTrans <- rnorm(1,zt+1,transition_sd)
  }
  else{
    normTrans <- rnorm(1,zt+2,transition_sd)
  }
  return(normTrans)
}

emmisionmodel <- function(zt,emission_sd){
  probs = rep(1/3,3)
```

```
  draw = sample(1:3,1,prob = probs)

  if(draw==1){
    normEmis <- rnorm(1,zt,emission_sd)
  }
  else if(draw==2){
    normEmis <- rnorm(1,zt-1,emission_sd)
  }
  else{
    normEmis <- rnorm(1,zt+1,emission_sd)
  }
  return(normEmis)
}

emission_density <- function(xt,zt){
  x <- (dnorm(xt,zt,emission_sd)+dnorm(xt,zt-1,emission_sd)+dnorm(xt,zt+1,emission_sd))/3
  return(x)
}


emission_sd = 1
transition_sd = 1

ssm <- function(emission_sd,transition_sd){
T = 100
particles = 100
zt = xt = error = rep(NA,T)

# for zt and xt
zt[1] = initModel(1)
for(i in 2:T){
  zt[i] <- transitionmodel(zt[i-1])
  xt[i] <- emmisionmodel(zt[i],emission_sd)
}

initParticles <- initModel(particles)
particleWeights<- rep(1/particles,particles)
estimation <- c()
Particles25 = Particles75 = NA

for(i in 2:T){
  newParticles <- sample(1:particles,particles,prob=particleWeights,replace = T)
  initParticles <- sapply(initParticles[newParticles],transitionmodel)

  if(i == 25){
    Particles25 <- c(initParticles)
  }
  else if(i == 75){
    Particles75 <- c(initParticles)
  }


  for(j in 1:particles){
```

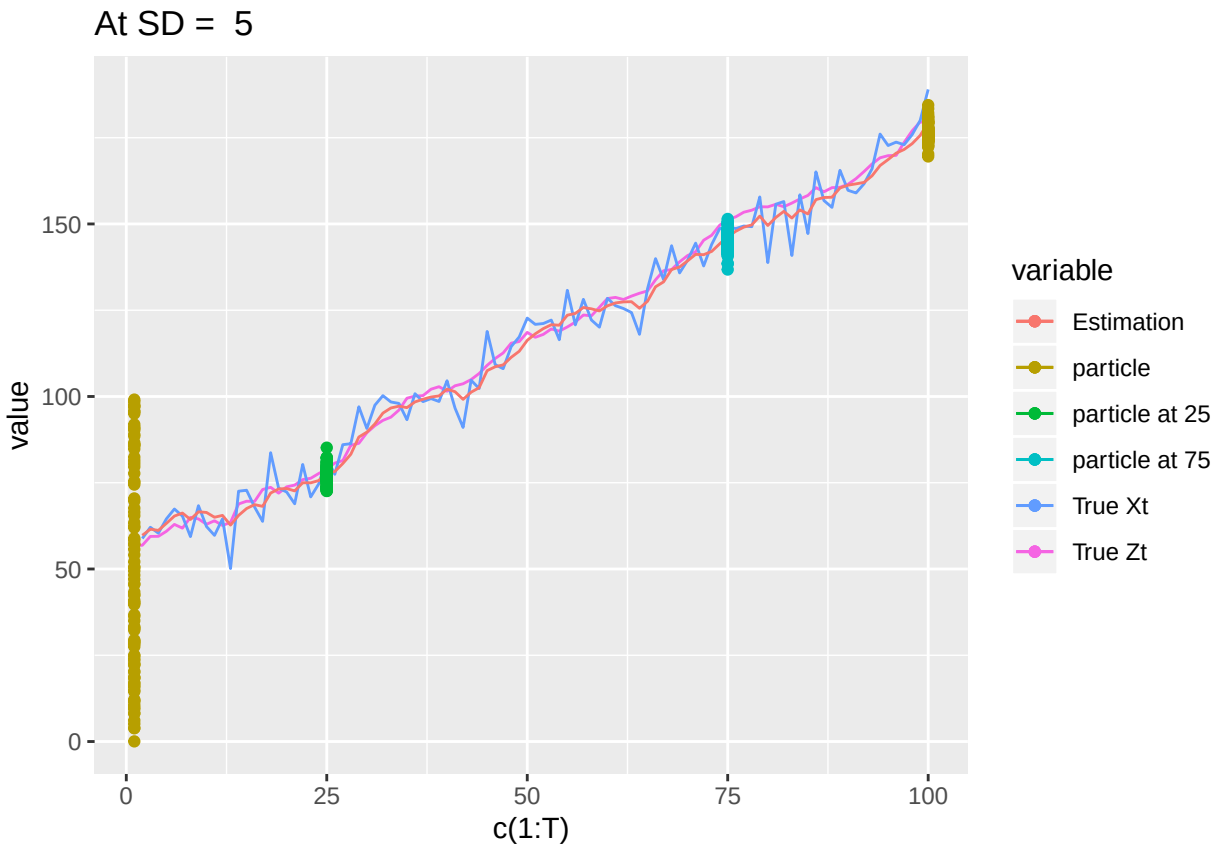
```
    particleWeights[j] <- emission_density(xt[i],initParticles[j])
  }
  #Normalise
  particleWeights <- particleWeights/sum(particleWeights)

  Ezt <- sum(particleWeights*initParticles)
  error[i] <- abs(zt[i]-Ezt)
  estimation[i] <- sum(particleWeights * initParticles)
}

data = data.frame(zt,xt,estimation,particles = initModel(particles),Particles25,Particles75,initParticl

ggplot(data,aes(x=c(1:T),y=value,color=variable,xlab="timestep"))+
  geom_line(aes(y=data$zt,col='True Zt'))+
  geom_line(aes(y=data$xt,col='True Xt'))+
  geom_line(aes(y=data$estimation,col='Estimation'))+
  geom_point(aes(x=rep(1,T),y=data$particles,col='particle'))+
  geom_point(aes(x=rep(25,T),y=data$Particles25,col='particle at 25'))+
  geom_point(aes(x=rep(75,T),y=data$Particles75,col='particle at 75'))+
  geom_point(aes(x=rep(T,T),y=data$initParticles,col='particle'))+
  ggtitle(paste('At SD = ',emission_sd))
}
ssm(emission_sd ,transition_sd )
```



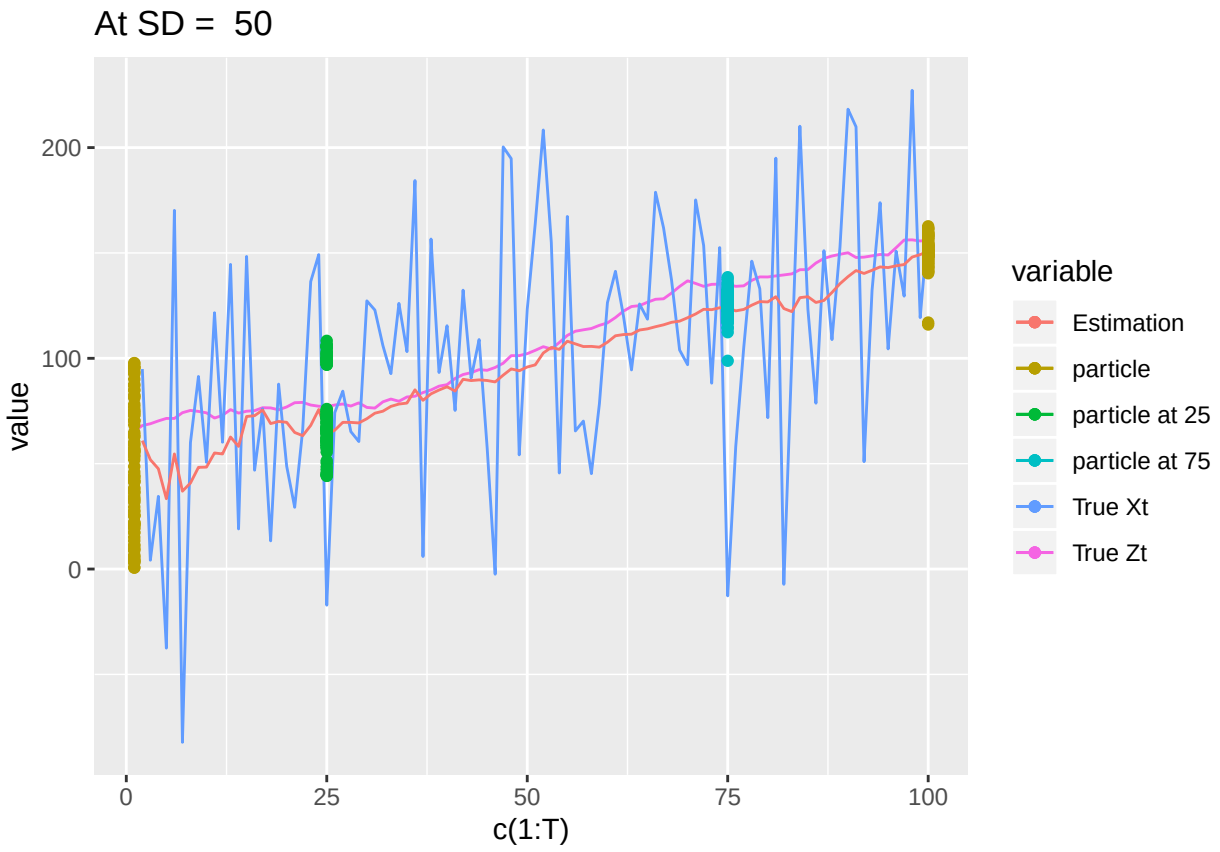
Intermediate particles number 25 and 75 are chosen and their locations are mapped in the figure.

B)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.

```
emission_sd = 5
ssm(emission_sd ,transition_sd )
```




```
emission_sd = 50
ssm(emission_sd ,transition_sd )
```

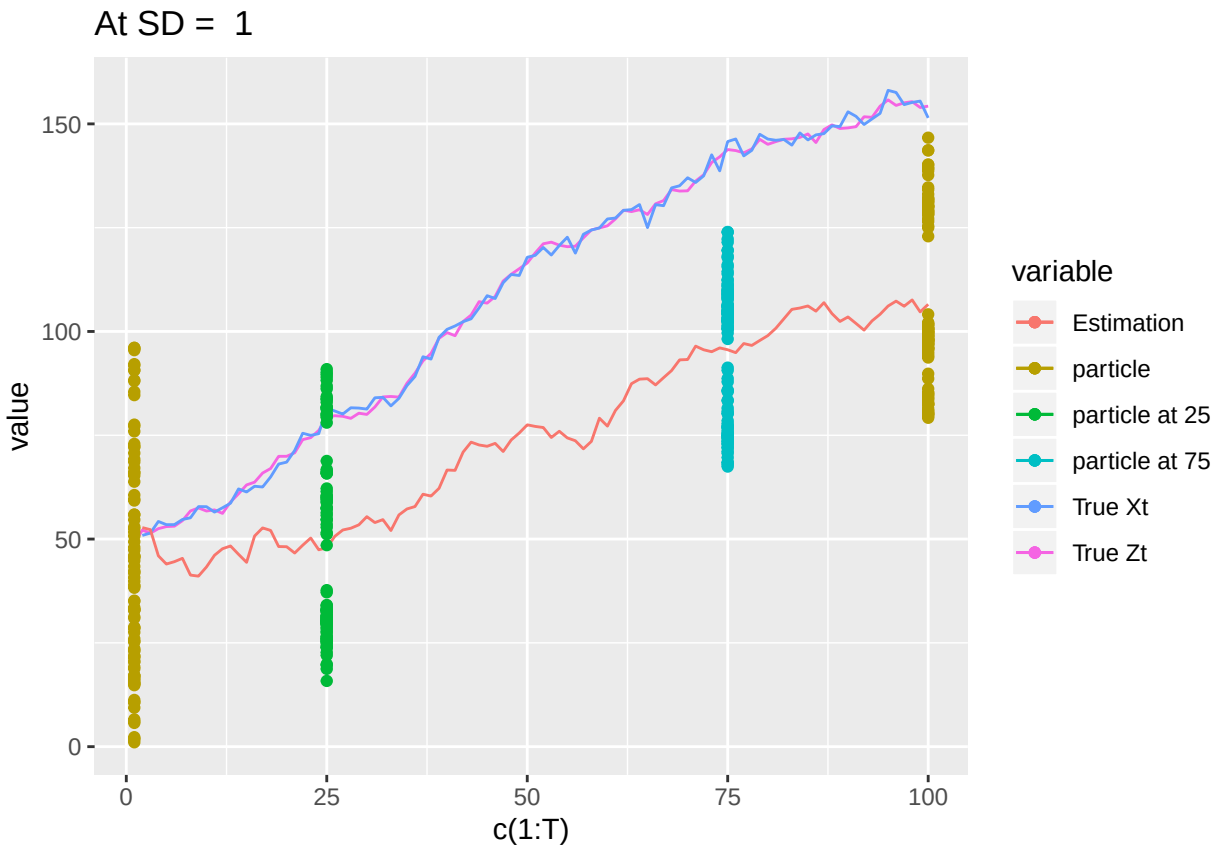


Increasing the standard deviation not only makes the true x_t scattered but also scatters the particles both intermediate and endstep.

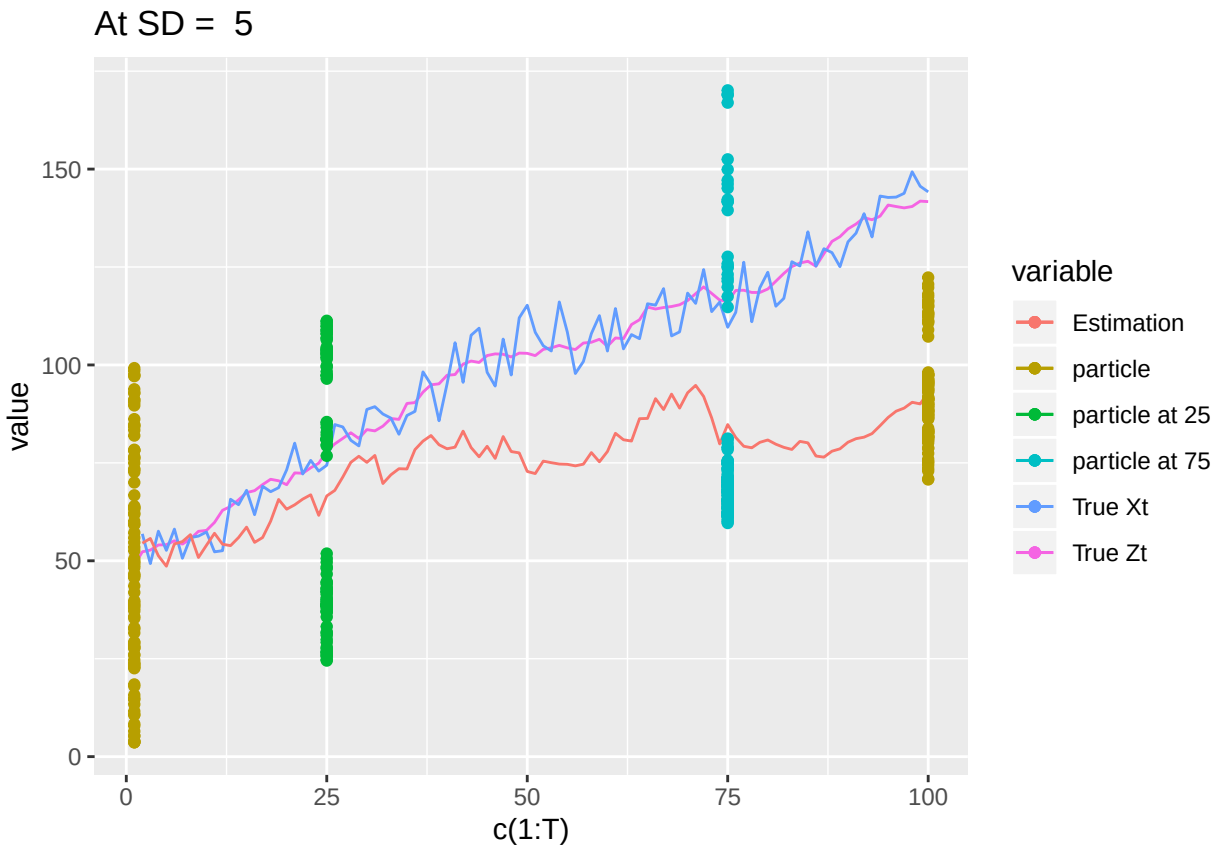
C) Finally, show and explain what happens when the weights in the particle filter are always equal to 1, i.e. there is no correction.

```
ssm <- function(emission_sd,transition_sd){
T = 100
particles = 100
zt = xt = error = rep(NA,T)

# for zt and xt
zt[1] = initModel(1)
for(i in 2:T){
  zt[i] <- transitionmodel(zt[i-1])
  xt[i] <- emmisionmodel(zt[i],emission_sd)
}

initParticles <- initModel(particles)
particleWeights<- rep(1/particles,particles)
estimation <- c()
Particles25 = Particles75 = NA

for(i in 2:T){
```

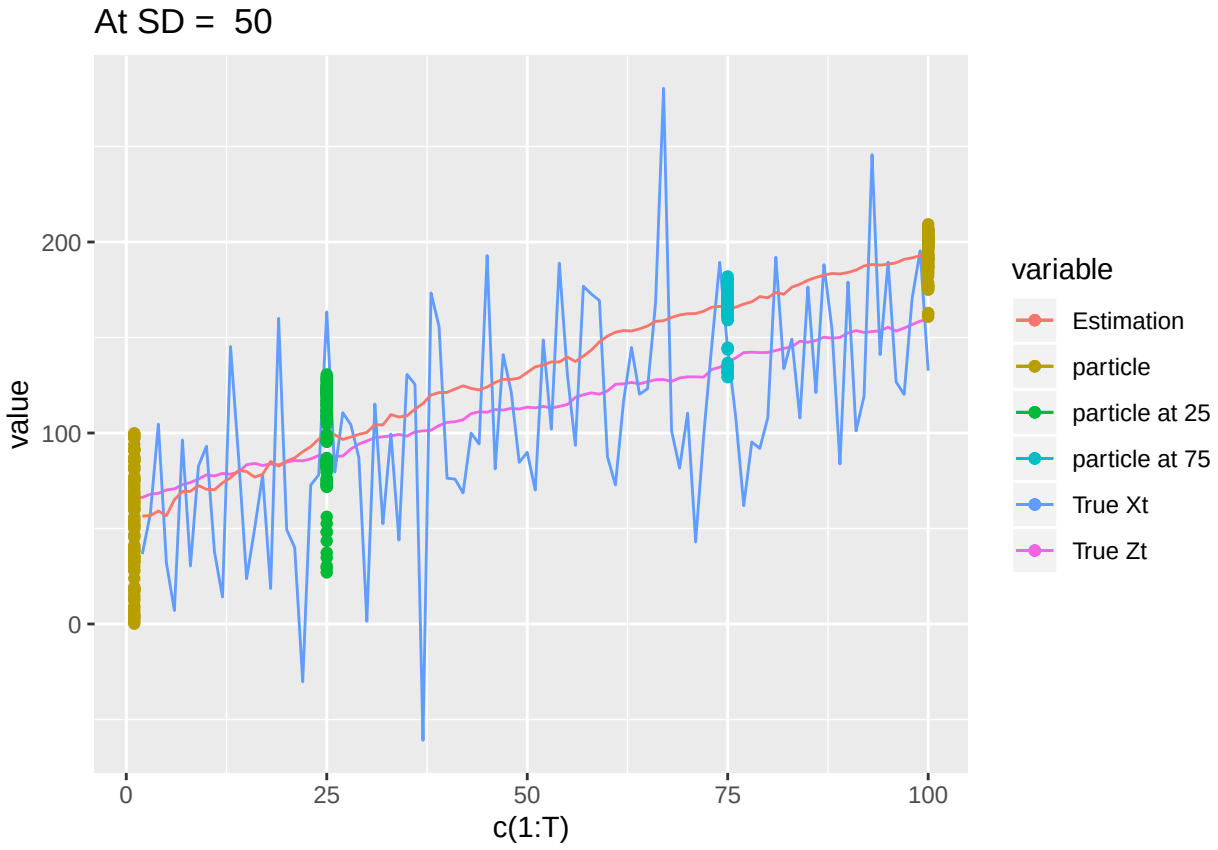
```
  newParticles <- sample(1:particles,particles,prob=particleWeights,replace = T)
  initParticles <- sapply(initParticles[newParticles],transitionmodel)

  if(i == 25){
    Particles25 <- c(initParticles)
  }
  else if(i == 75){
    Particles75 <- c(initParticles)
  }


  for(j in 1:particles){
    particleWeights[j] <- 1
  }
  #Normalise
  particleWeights <- particleWeights/sum(particleWeights)

  Ezt <- sum(particleWeights*initParticles)
  error[i] <- abs(zt[i]-Ezt)
  estimation[i] <- sum(particleWeights * initParticles)
}

data = data.frame(zt,xt,estimation,particles = initModel(particles),Particles25,Particles75,initParticl

ggplot(data,aes(x=c(1:T),y=value,color=variable,xlab="timestep"))+
  geom_line(aes(y=data$zt,col='True Zt'))+
  geom_line(aes(y=data$xt,col='True Xt'))+
  geom_line(aes(y=data$estimation,col='Estimation'))+
  geom_point(aes(x=rep(1,T),y=data$particles,col='particle'))+
  geom_point(aes(x=rep(25,T),y=data$Particles25,col='particle at 25'))+
  geom_point(aes(x=rep(75,T),y=data$Particles75,col='particle at 75'))+
  geom_point(aes(x=rep(T,T),y=data$initParticles,col='particle'))+
  ggtitle(paste('At SD = ',emission_sd))
}

emission_sd = 1
ssm(emission_sd ,transition_sd )
```




```
emission_sd = 5
ssm(emission_sd ,transition_sd )
```




```
emission_sd = 50
ssm(emission_sd ,transition_sd )
```



It is seen that the estimation and true values do not match and accuracy is reduced. This change is consistent from all the standard deviation runs of 1,5 and 50. value of the particle at all timesteps are scattered as compared to earlier runs. It reduces the quality of the filtering.

```
knitr::opts_chunk$set(echo = TRUE)
library("dplyr")
library("ggplot2")
library("KFAS")
initModel <- function(len){
  x <- runif(len,0,100)
  return(x)
}

transitionmodel <- function(zt){
  probs = rep(1/3,3)
  draw = sample(1:3,1,prob = probs)
  if(draw==1){
    normTrans <- rnorm(1,zt,transition_sd)
  }
  else if(draw==2){
    normTrans <- rnorm(1,zt+1,transition_sd)
  }
  else{
    normTrans <- rnorm(1,zt+2,transition_sd)
  }
  return(normTrans)
```

```
}

emmisionmodel <- function(zt,emission_sd){
  probs = rep(1/3,3)
  draw = sample(1:3,1,prob = probs)

  if(draw==1){
    normEmis <- rnorm(1,zt,emission_sd)
  }
  else if(draw==2){
    normEmis <- rnorm(1,zt-1,emission_sd)
  }
  else{
    normEmis <- rnorm(1,zt+1,emission_sd)
  }
  return(normEmis)
}

emission_density <- function(xt,zt){
  x <- (dnorm(xt,zt,emission_sd)+dnorm(xt,zt-1,emission_sd)+dnorm(xt,zt+1,emission_sd))/3
  return(x)
}


emission_sd = 1
transition_sd = 1

ssm <- function(emission_sd,transition_sd){
T = 100
particles = 100
zt = xt = error = rep(NA,T)

# for zt and xt
zt[1] = initModel(1)
for(i in 2:T){
  zt[i] <- transitionmodel(zt[i-1])
  xt[i] <- emmisionmodel(zt[i],emission_sd)
}

initParticles <- initModel(particles)
particleWeights<- rep(1/particles,particles)
estimation <- c()
Particles25 = Particles75 = NA

for(i in 2:T){
  newParticles <- sample(1:particles,particles,prob=particleWeights,replace = T)
  initParticles <- sapply(initParticles[newParticles],transitionmodel)

  if(i == 25){
    Particles25 <- c(initParticles)
  }
  else if(i == 75){
    Particles75 <- c(initParticles)
```

```
  }


  for(j in 1:particles){
    particleWeights[j] <- emission_density(xt[i],initParticles[j])
  }
  #Normalise
  particleWeights <- particleWeights/sum(particleWeights)

  Ezt <- sum(particleWeights*initParticles)
  error[i] <- abs(zt[i]-Ezt)
  estimation[i] <- sum(particleWeights * initParticles)
}

data = data.frame(zt,xt,estimation,particles = initModel(particles),Particles25,Particles75,initParticl

ggplot(data,aes(x=c(1:T),y=value,color=variable,xlab="timestep"))+
  geom_line(aes(y=data$zt,col='True Zt'))+
  geom_line(aes(y=data$xt,col='True Xt'))+
  geom_line(aes(y=data$estimation,col='Estimation'))+
  geom_point(aes(x=rep(1,T),y=data$particles,col='particle'))+
  geom_point(aes(x=rep(25,T),y=data$Particles25,col='particle at 25'))+
  geom_point(aes(x=rep(75,T),y=data$Particles75,col='particle at 75'))+
  geom_point(aes(x=rep(T,T),y=data$initParticles,col='particle'))+
  ggtitle(paste('At SD = ',emission_sd))
}
ssm(emission_sd ,transition_sd )
emission_sd = 5
ssm(emission_sd ,transition_sd )

emission_sd = 50
ssm(emission_sd ,transition_sd )
ssm <- function(emission_sd,transition_sd){
T = 100
particles = 100
zt = xt = error = rep(NA,T)

# for zt and xt
zt[1] = initModel(1)
for(i in 2:T){
  zt[i] <- transitionmodel(zt[i-1])
  xt[i] <- emmisionmodel(zt[i],emission_sd)
}

initParticles <- initModel(particles)
particleWeights<- rep(1/particles,particles)
estimation <- c()
Particles25 = Particles75 = NA

for(i in 2:T){
  newParticles <- sample(1:particles,particles,prob=particleWeights,replace = T)
  initParticles <- sapply(initParticles[newParticles],transitionmodel)
```

```
  if(i == 25){
    Particles25 <- c(initParticles)
  }
  else if(i == 75){
    Particles75 <- c(initParticles)
  }


  for(j in 1:particles){
    particleWeights[j] <- 1
  }
  #Normalise
  particleWeights <- particleWeights/sum(particleWeights)

  Ezt <- sum(particleWeights*initParticles)
  error[i] <- abs(zt[i]-Ezt)
  estimation[i] <- sum(particleWeights * initParticles)
}

data = data.frame(zt,xt,estimation,particles = initModel(particles),Particles25,Particles75,initParticl

ggplot(data,aes(x=c(1:T),y=value,color=variable,xlab="timestep"))+
  geom_line(aes(y=data$zt,col='True Zt'))+
  geom_line(aes(y=data$xt,col='True Xt'))+
  geom_line(aes(y=data$estimation,col='Estimation'))+
  geom_point(aes(x=rep(1,T),y=data$particles,col='particle'))+
  geom_point(aes(x=rep(25,T),y=data$Particles25,col='particle at 25'))+
  geom_point(aes(x=rep(75,T),y=data$Particles75,col='particle at 75'))+
  geom_point(aes(x=rep(T,T),y=data$initParticles,col='particle'))+
  ggtitle(paste('At SD = ',emission_sd))
}

emission_sd = 1
ssm(emission_sd ,transition_sd )

emission_sd = 5
ssm(emission_sd ,transition_sd )

emission_sd = 50
ssm(emission_sd ,transition_sd )
# Question 4: SSMs
# Kalman filter implementation.
set.seed(12345)
start_time <- Sys.time()

T<-10000
mu_0<-50
Sigma_0<-10
R<-1
Q<-5

x<-vector(length=T)
z<-vector(length=T)
```

```r
err<-vector(length=T)

for(t in 1:T){
  x[t]<-ifelse(t==1,rnorm(1,mu_0,Sigma_0),x[t-1]+1+rnorm(1,0,R))
  z[t]<-x[t]+rnorm(1,0,Q)
}

mu<-mu_0
Sigma<-Sigma_0*Sigma_0 # KF uses covariances
for(t in 2:T){
  pre_mu<-mu+1
  pre_Sigma<-Sigma+R*R # KF uses covariances
  K<-pre_Sigma/(pre_Sigma+Q*Q) # KF uses covariances
  mu<-pre_mu+K*(z[t]-pre_mu)
  Sigma<-(1-K)*pre_Sigma

  err[t]<-abs(x[t]-mu)

  cat("t: ",t,", x_t: ",x[t],", E[x_t]: ",mu," , error: ",err[t],"\n")
  flush.console()
}

mean(err[2:T])
sd(err[2:T])

end_time <- Sys.time()
end_time - start_time

# Repetition with the particle filter.

set.seed(12345)
start_time <- Sys.time()

T<-10000
n_par<-100
tra_sd<-1
emi_sd<-5
mu_0<-50
Sigma_0<-10

ini_dis<-function(n){
  return (rnorm(n,mu_0,Sigma_0))
}

tra_dis<-function(zt){
  return (rnorm(1,mean=zt+1,sd=tra_sd))
}

emi_dis<-function(zt){
  return (rnorm(1,mean=zt,sd=emi_sd))
}

den_emi_dis<-function(xt,zt){
```

```
  return (dnorm(xt,mean=zt,sd=emi_sd))
}

z<-vector(length=T)
x<-vector(length=T)

for(t in 1:T){
  z[t]<-ifelse(t==1,ini_dis(1),tra_dis(z[t-1]))
  x[t]<-emi_dis(z[t])
}

err<-vector(length=T)

bel<-ini_dis(n_par)
w<-rep(1/n_par,n_par)
for(t in 2:T){
  com<-sample(1:n_par,n_par,replace=TRUE,prob=w)
  bel<-sapply(bel[com],tra_dis)

  for(i in 1:n_par){
    w[i]<-den_emi_dis(x[t],bel[i])
  }
  w<-w/sum(w)

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

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

mean(err[2:T])
sd(err[2:T])

end_time <- Sys.time()
end_time - start_time

# KF works optimally (i.e. it computes the exact belief function in closed-form) since the SSM sampled
# linear-Gaussian. The particle filter on the other hand is approximate. The more particles the closer
# performance to the KF's but at the cost of increasing the running time.
# SSMs
set.seed(12345)
start_time <- Sys.time()

T<-100
mu_0<-50
Sigma_0<-10
R<-1
Q<-5

x<-vector(length=T)
z<-vector(length=T)
err<-vector(length=T)
```

```
for(t in 1:T){
  x[t]<-ifelse(t==1,rnorm(1,mu_0,Sigma_0),x[t-1]+1+rnorm(1,0,R))
  z[t]<-x[t]+rnorm(1,0,Q)
}

mu<-mu_0
Sigma<-Sigma_0*Sigma_0 # KF uses covariances
for(t in 2:T){
  pre_mu<-mu+1
  pre_Sigma<-Sigma+R*R # KF uses covariances
  K<-pre_Sigma/(pre_Sigma+Q*Q) # KF uses covariances
  mu<-pre_mu+K*(z[t]-pre_mu)
  Sigma<-(1-K)*pre_Sigma

  err[t]<-abs(x[t]-mu)

  cat("t: ",t,", x_t: ",x[t],", E[x_t]: ",mu," , error: ",err[t],"\n")
  flush.console()
}

mean(err[2:T])
sd(err[2:T])

end_time <- Sys.time()
end_time - start_time
```