AML_Lab3_Hyungyum_Kim

Josh Hyungyum Kim

2018 - 10 - 4

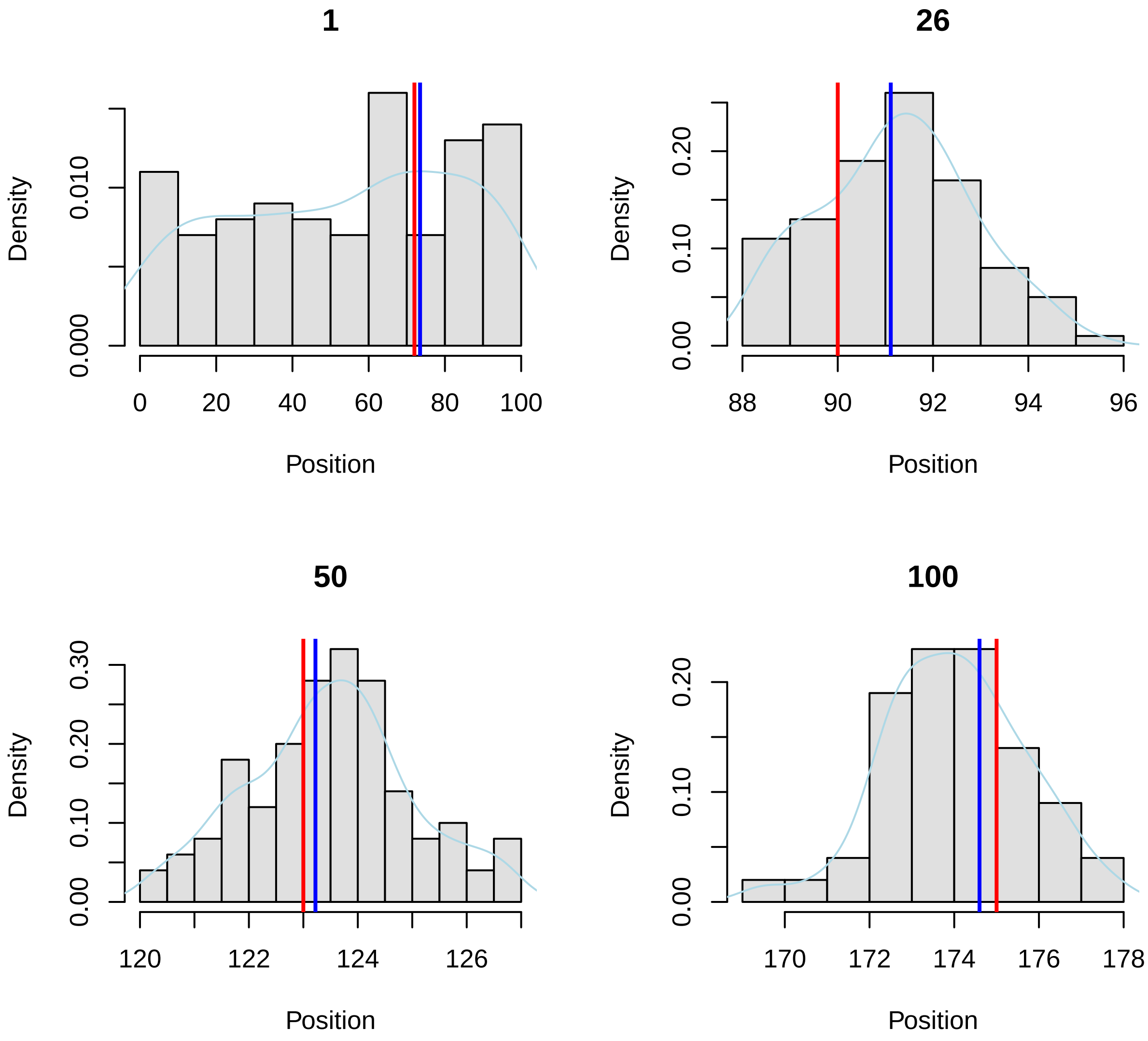
1. Implement the SSM.

```r
# 1.

# seed for reproducible results
set.seed(12345)

# T = 100
Tsteps <- 100

# Vectors for true states and observations
Tstates <- c()
obs <- c()

# Initial model
init <- round(runif(1, 0, 100))

# simulation of 100 time steps
for(i in 1:Tsteps){

  if (i==1){
    Tstates[1] <- init
  }else{
    # select one model of mixtures
    mix_comp2 <- sample((0:2),1)
    Tstates[i] <- round(rnorm(1, Tstates[i-1]+mix_comp2, 1))
  }

  # select one model of mixtures
  mix_comp <- sample((-1:1),1)
  # observation
  obs[i] <- round(rnorm(1, Tstates[i]+mix_comp, 1))
}

# Print the results
list("True_location"=Tstates, "Expected_location"=obs)
```

```
## $True_location
##   [1]  72  72  71  72  75  75  76  78  76  75  75  76  78  80  78  78  80
##  [18]  80  82  84  84  87  87  88  88  90  91  93  94  95  98 101 104 105
##  [35] 105 106 108 107 109 111 112 113 115 114 117 119 121 122 123 123 122
##  [52] 123 124 125 126 125 123 125 129 131 132 134 136 139 139 139 138 138
##  [69] 139 137 137 139 138 139 139 143 145 147 148 150 150 151 154 157 161
##  [86] 163 163 164 164 165 165 167 169 168 169 170 170 172 174 175
##
## $Expected_location
##   [1]  74  73  70  73  76  73  77  78  78  76  76  75  76  80  78  78  80
##  [18]  80  81  87  83  85  89  89  91  91  92  94  96  97  97 101 105 105
##  [35] 105 107 108 108 108 113 114 113 115 116 115 117 120 120 124 123 123
##  [52] 122 123 126 127 127 124 126 130 132 131 134 136 140 140 141 138 137
##  [69] 138 138 135 137 135 139 140 143 144 148 147 150 148 152 154 157 161
##  [86] 163 161 164 165 165 163 167 170 166 170 170 170 172 173 175
```

The SSM described in lab instruction is implemented and the simulation of 100 time steps are listed above. The observations will be used to identify the state via particle filtering(100 particles). Particle filtering algorithm is implemented as below:

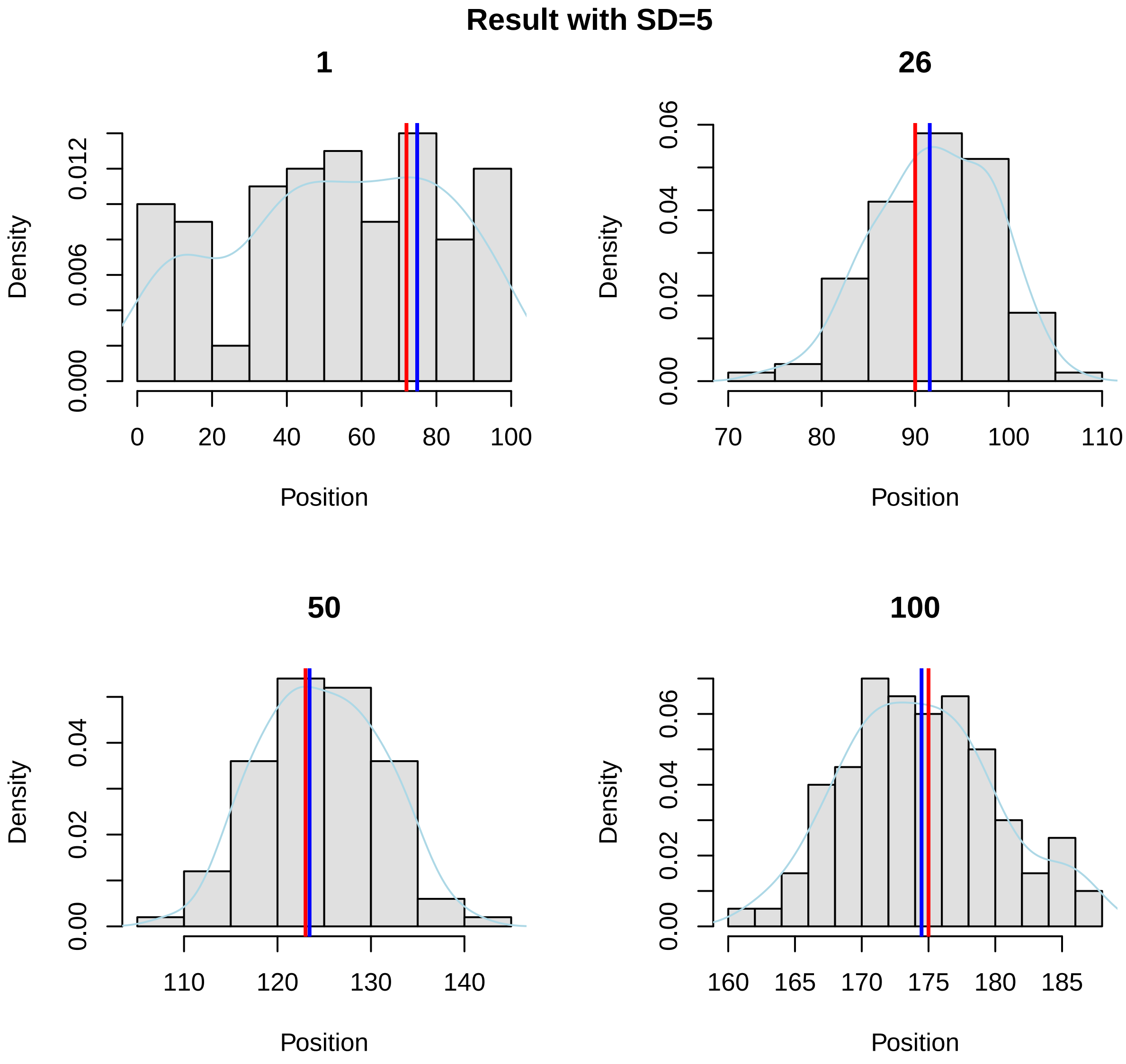
```r
# particle filter

# emission model
emi_model <- function(z, sd){
  # select one model of mixtures
  mix_comp <- sample((-1:1), 1)

  rnorm(1, z+mix_comp, sd)
}

# transition model
trans_model <- function(z, sd){
  iter <- length(z)
  res <- c()

  for(i in 1:iter){
    mix_comp <- sample((0:2), 1)
    res[i] <- rnorm(1, z[i]+mix_comp, sd)
  }
  res
}

# density function
den_fun <- function(mean, input, sd){
  (dnorm(x=input, mean=mean, sd=sd)+
   dnorm(x=input, mean=mean+1, sd=sd)+
   dnorm(x=input, mean=mean-1, sd=sd))/3
}

# 100 particles
num_parti <- 100

particle <- function(weight=TRUE, sd=1){

  # lists for storing weights and particles
  Zn <- list()
  Wn <- list()

  # 100 samples
  Zn[[1]] <- runif(num_parti, 0 ,100)

  for(i in 1:Tsteps){

    numer <- sapply(X = Zn[[i]], FUN = den_fun, input=obs[i], sd=sd)
    Wn[[i]] <- numer / sum(numer)
    if(i==Tsteps){break}

    # correction and prediction
    if(weight==TRUE){
      ind <- sample(1:num_parti, 100, replace=TRUE, prob=Wn[[i]])
    }else{
      ind <- sample(1:num_parti, 100, replace=TRUE)
    }

    # new particle
    Zn[[i+1]] <- trans_model(Zn[[i]][ind], sd=sd)
  }
  list("Weights"=Wn, "Particles"=Zn)
}

res <- particle()

parti_plot <- function(res){
  Wn <- res$Weights
  Zn <- res$Particles

  par(mfrow=c(2,2))
  for(i in c(1,26,50,100)){
    hist(Zn[[i]], freq = FALSE, col="grey88", breaks=10,
         main=i, xlab="Position")
    lines(density(Zn[[i]]), col="lightblue")
    abline(v=sum(Wn[[i]]*Zn[[i]]), col="blue", lwd=2) # Prediction
    abline(v=Tstates[i], col="Red", lwd=2) # True State
  }
}

parti_plot(res)
```



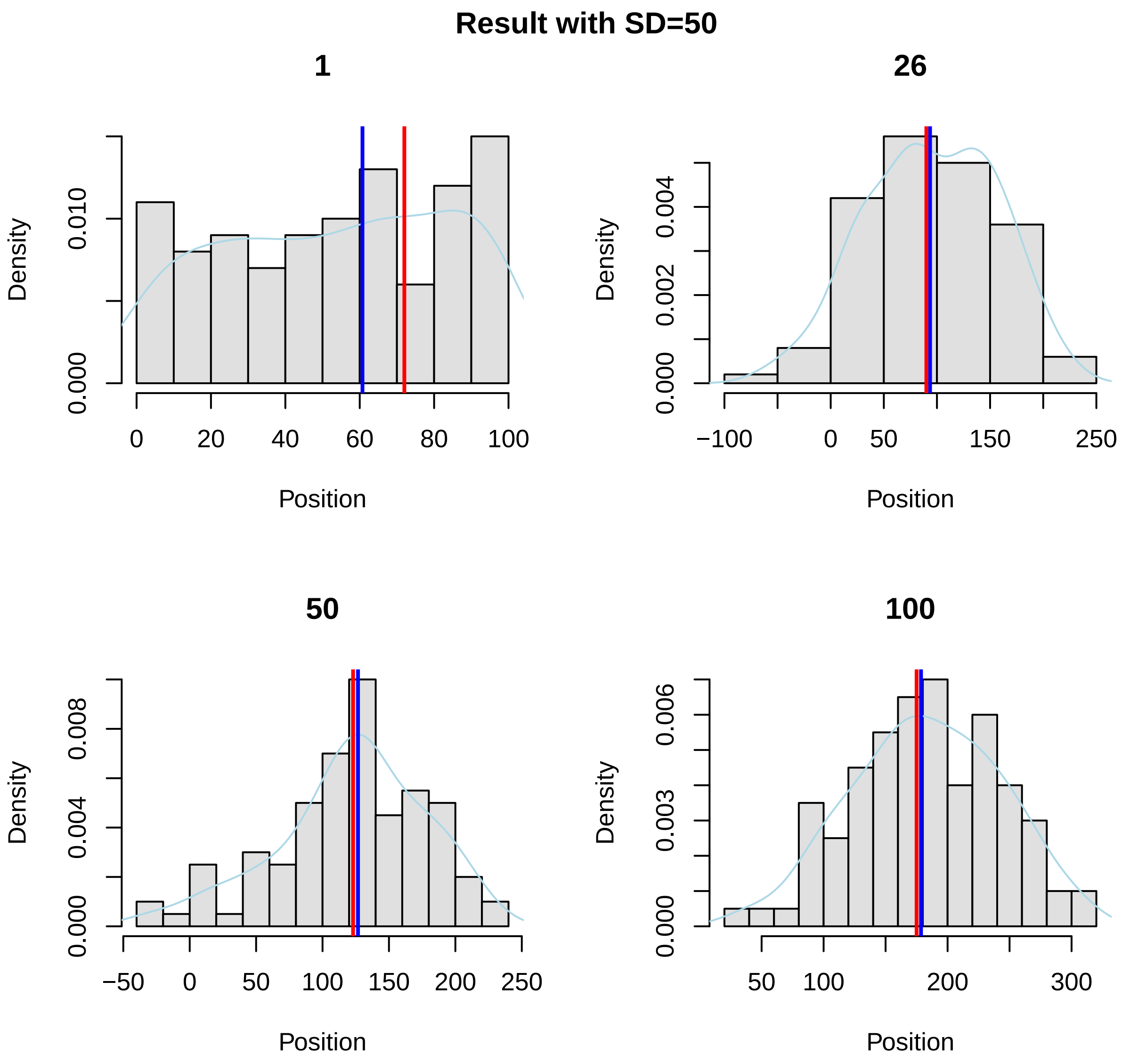
Above 4 different histograms show particles of time step 1, 26, 50 and 100. Lightblue line means approximated density, blue line means expected location and the red line means the true location.

2. Repeat with SD=5, 50.

```
# 2.

# result with SD=5
res5 <- particle(sd=5)
parti_plot(res5)
title("Result with SD=5", outer=TRUE, line=-1)
```



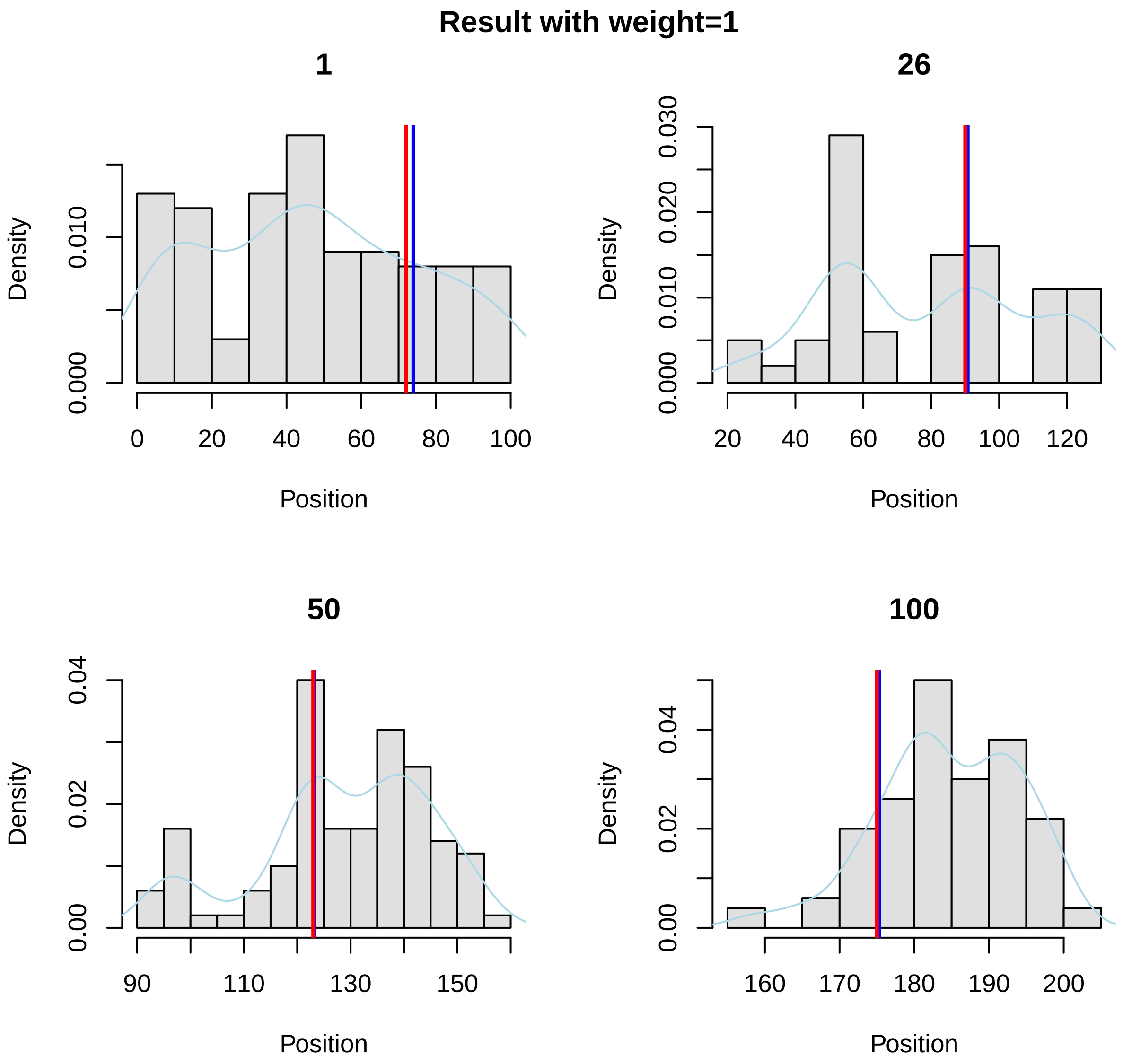
```
# result with SD=50
res50 <- particle(sd=50)
parti_plot(res50)
title("Result with SD=50", outer=TRUE, line=-1)
```



Based on the above plots, one might say that it is obvious the particles become more and more sparsed when the sd increases. This is a natural outcome since the sd influence the probability distributions of emission model and transition model. However, the result of expected locations is still quite close to the true locations.

3. Weight=1.

```
# 3.

res_nofil <- particle(weight = FALSE)
parti_plot(res_nofil)
title("Result with weight=1", outer=TRUE, line=-1)
```



Even though there is no weighting to the particles, the result of expected locations are pretty nice.