AML_Lab3_Hyungyum_Kim

Josh Hyungyum Kim

2018 - 10 - 4

1. Implement the SSM.

```
# 1.

# T = 100
Tsteps <- 100

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

# Initial model
set.seed(123456) # seed for reproducible results
init <- round(runif(1, 0, 100))

# simulation of 100 time steps
set.seed(123456)
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)
```

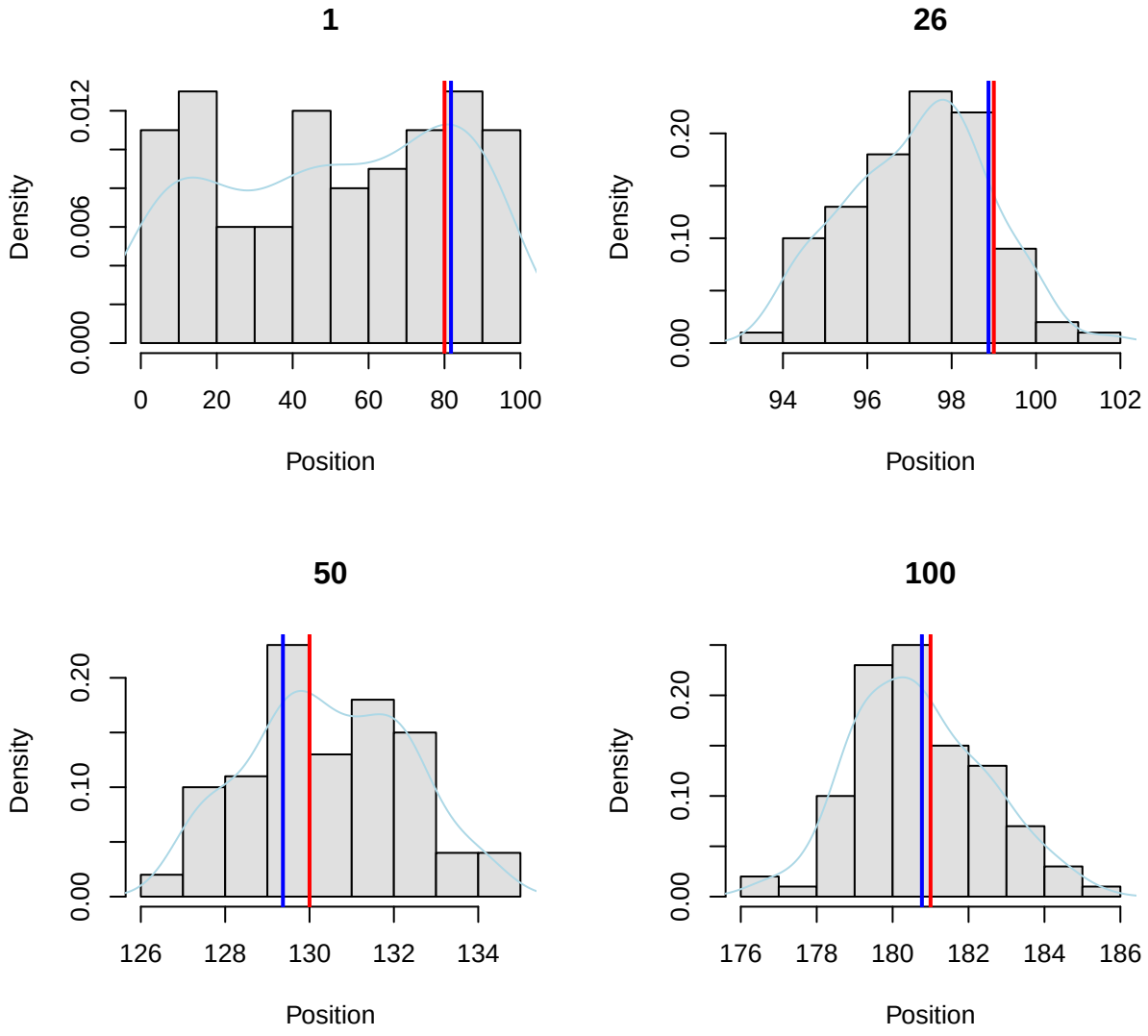
```
## $True_location
##   [1]  80  81  82  85  84  87  88  88  89  91  92  91  90  91  92  93  93
##  [18]  92  94  93  93  95  95  96  98  99 101 102 104 105 105 108 110 112
##  [35] 113 114 116 116 118 118 119 118 121 120 121 120 123 127 128 130 131
##  [52] 131 134 136 138 139 142 141 142 142 143 144 146 146 147 147 148 147
##  [69] 148 147 149 151 152 154 155 156 157 160 161 161 161 162 164 164 163
##  [86] 163 164 164 167 167 168 167 171 169 171 174 177 178 180 181
##
## $Expected_location
##   [1]  82  80  84  86  85  88  90  88  90  94  89  90  91  90  92  92  95
##  [18]  90  95  92  91  92  95  93  97 100 101 102 102 108 104 108 110 112
```

```
##  [35] 112 114 114 116 116 118 119 117 120 120 123 121 123 127 131 129 129
##  [52] 131 134 137 140 140 141 142 143 141 142 145 144 145 147 146 147 147
##  [69] 147 148 147 151 156 156 154 156 156 159 161 163 161 161 163 164 161
##  [86] 164 164 165 167 167 167 168 173 170 171 174 178 178 180 181
```

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

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

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

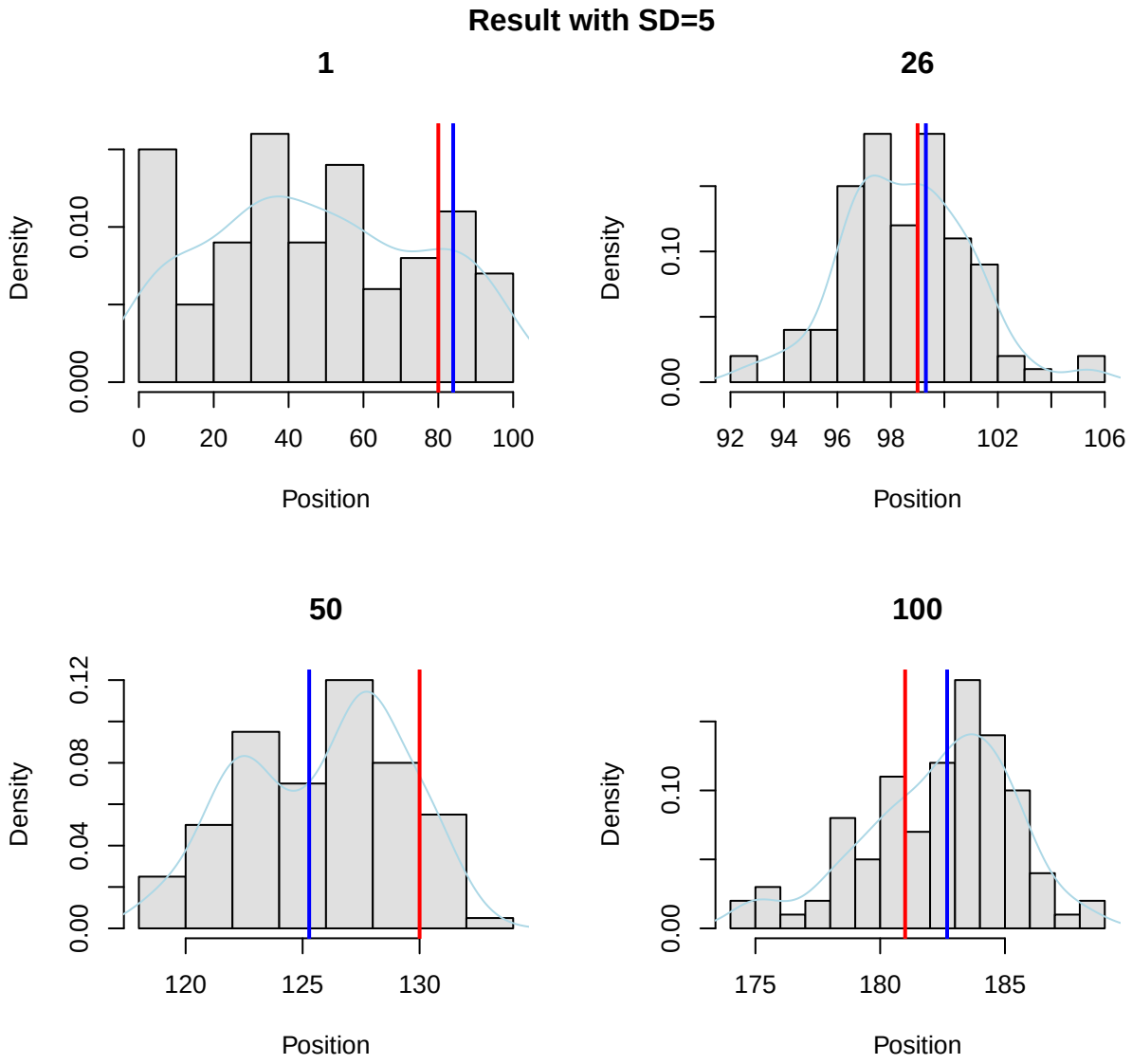
```
    # new particle
    Zn[[i+1]] <- trans_model(Zn[[i]][ind])
  }
  list("Weights"=Wn, "Particles"=Zn)
}

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)
```



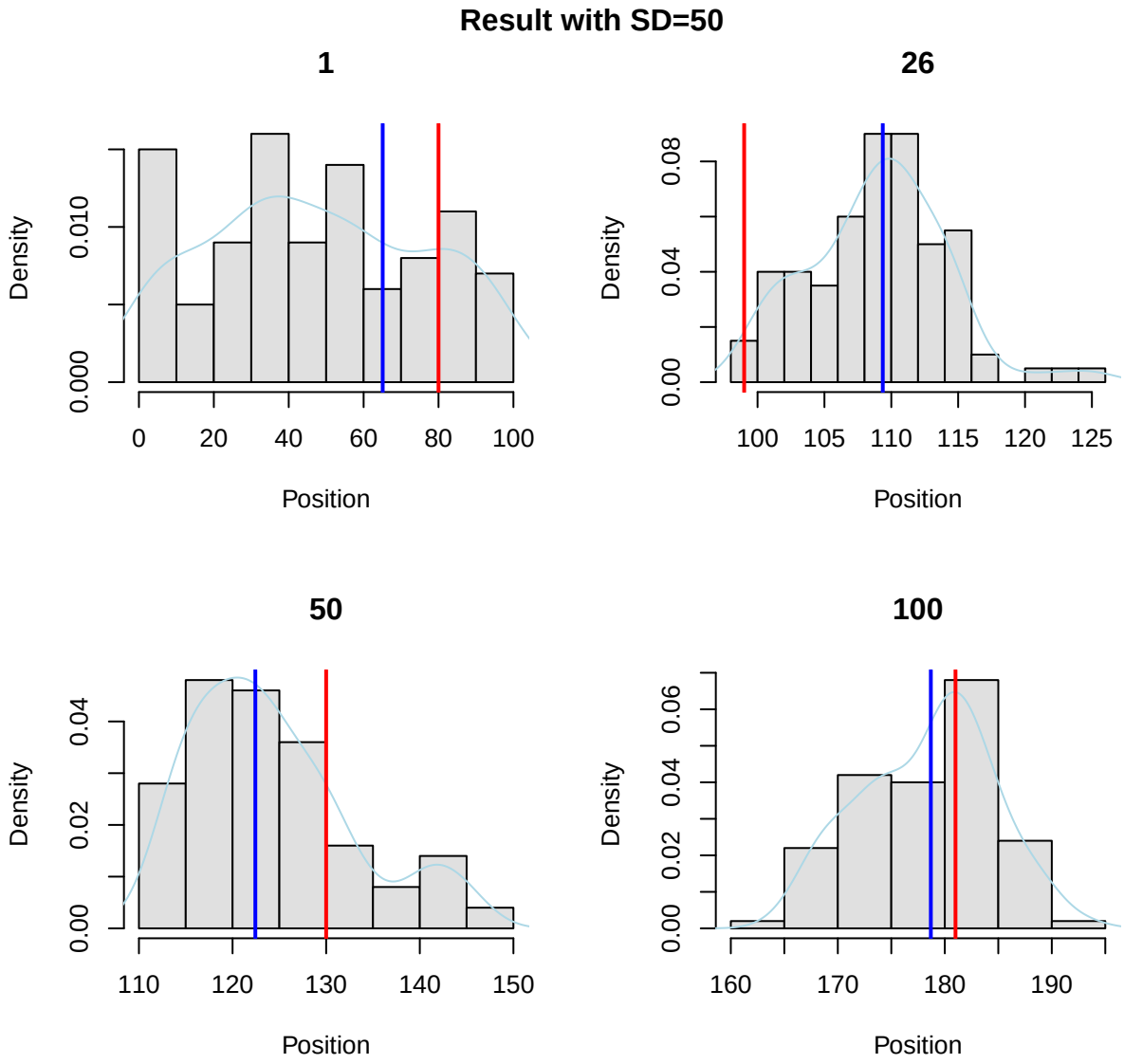
Above 4 different histograms show particles of time step 1, 26, 50 and 100. Lightblue line means approximated density, blue line means expected location and the red line means the true location.

2. Repeat with SD=5, 50.

```r
# 2.

# simulation of 100 time steps with sd=5
set.seed(123456)
for(i in 1:Tsteps){

  # select one model of mixtures
  mix_comp <- sample((-1:1),1)
  # observation
  obs[i] <- round(rnorm(1, Tstates[i]+mix_comp, 5))
}

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



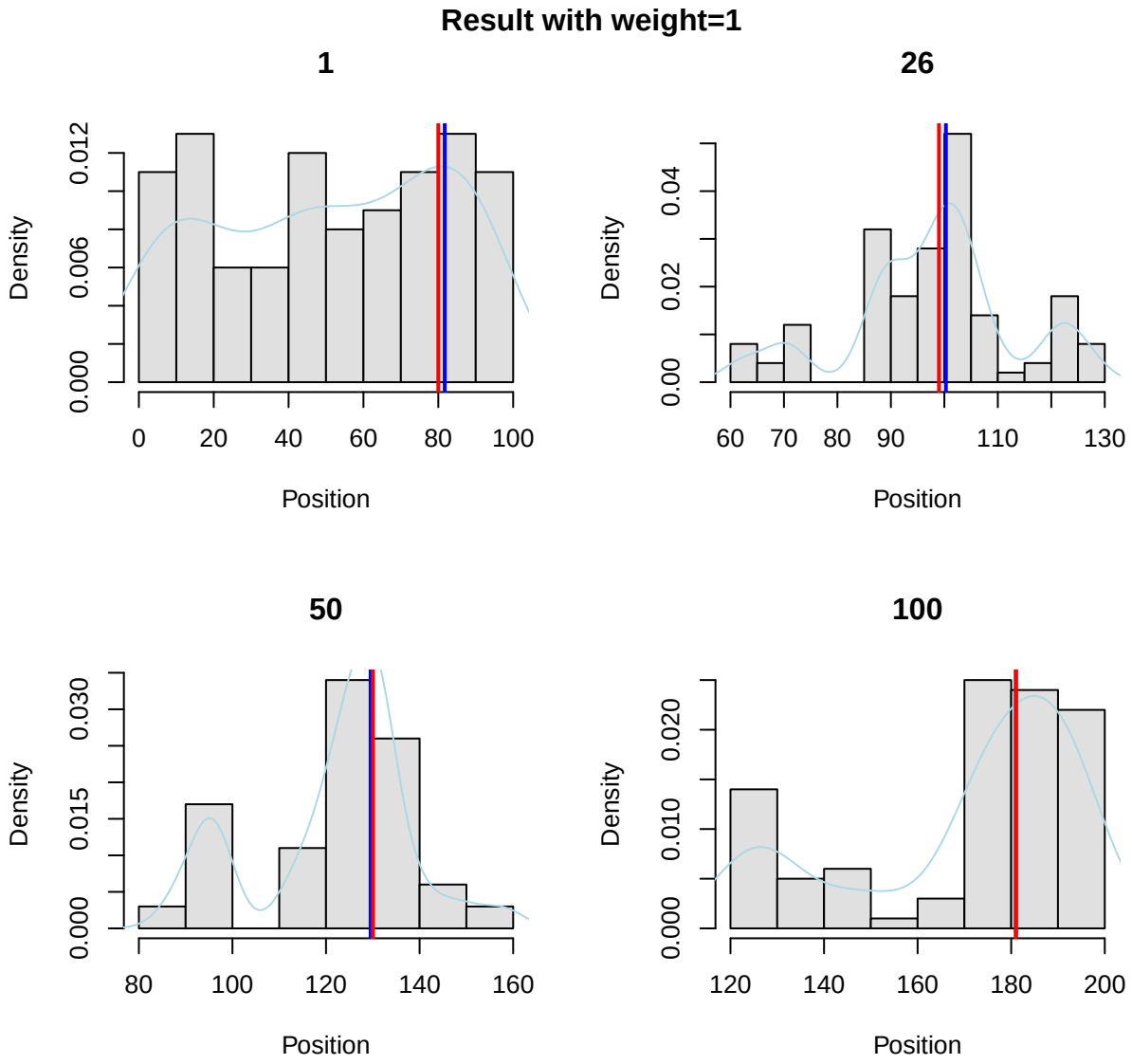

```
# simulation of 100 time steps with sd=50
set.seed(123456)
for(i in 1:Tsteps){

  # select one model of mixtures
  mix_comp <- sample((-1:1),1)
  # observation
  obs[i] <- round(rnorm(1, Tstates[i]+mix_comp, 50))
}

# 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. Furthermore, the results of prediction get worse since high standard deviation gives more uncertatinty for observations.

3. Weight=1.

```
# 3.

# To use same True state and obs for Q3 as Q1
set.seed(123456)
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))
}

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



When the weight=1 for every particle which means there is no correction, results of 4 time steps are really close to the true states. Maybe it is because the transition model is the same and the particles could move along with the observations. However, the histograms of particles are very sparsed than any weighted runs above.