Advanced Machine Learning (732A96) - Lab 03

Hector Plata (hecpl268)

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

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

Emission model

$$p\left(x_t|z_t\right) = \left(\mathcal{N}\left(x_t|z_t,1\right) + \mathcal{N}\left(x_t|z_t-1,1\right) + \mathcal{N}\left(x_t|z_t+1,1\right)\right)/3$$

Initial model

$$p\left(z_1\right) = Uniform(0,100)$$

Task 1

Implement the SMM above. Simulate it for $T = 100$ time steps to obtain $z_{1:100}$ (i.e., states) and $x_{1:100}$ (i.., 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.

Answer The figure can be seen below. The particles are can be seen as the density on the background of the figure. This means that all the figures are going to show the particles, the expected location and the true location for all of the time steps.

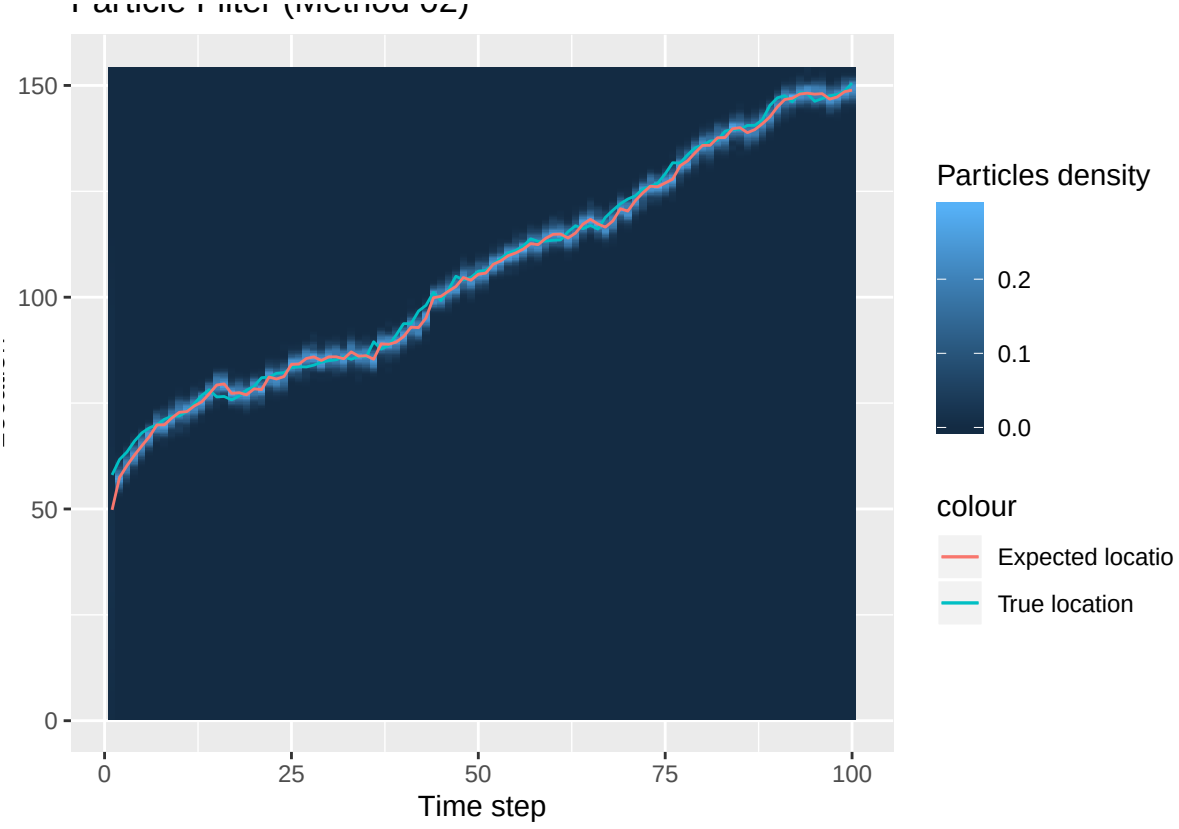
```
library(ggplot2)
library(reshape2)

set.seed(4)

# Sampling functions for the model.
sample_transition = function(z_prev, std=1, n=1)
{
  u = runif(1)

  if (u <= 1 / 3)
  {
    sample = rnorm(n, z_prev, std)
  }
  else if (u <= 2 /3)
```

```
  {
    sample = rnorm(n, z_prev + 1, std)
  }
  else
  {
    sample = rnorm(n, z_prev + 2, std)
  }

  return(sample)
}

sample_emission = function(z_actual, std=1, n=1)
{
  u = runif(1)

  if (u <= 1 / 3)
  {
    sample = rnorm(n, z_actual, std)
  }
  else if (u <= 2 / 3)
  {
    sample = rnorm(n, z_actual - 1, std)
  }
  else
  {
    sample = rnorm(n, z_actual - 1, std)
  }

  return(sample)
}

sample_init = function(n=1)
{
  return(runif(n, 0, 100))
}

density_emission = function(z_t, x_t, std=1)
{
  density =(dnorm(x_t, z_t, sd=std) +
            dnorm(x_t, z_t - 1, sd=std) +
            dnorm(x_t, z_t + 1, sd=std)) / 3

  return(density)
}

# Auxiliary plot functions.
give_me_plot = function(particle_res, sim_res, title="")
{
  # Getting the max and min values of the particles and true location.
  max_z = max(particle_res$Z, sim_res$Z)
  min_z = min(particle_res$Z, sim_res$Z)

  # Creating the data for the plots.
```

```
  time = c()
  position = c()
  densities = c()

  for (t in 1:n_iter)
  {
    particles_position_t = particle_res$Z[t, ]
    Z_density = density(particles_position_t, from=min_z, to=max_z)
    time = c(time, rep(t, 512))
    position = c(position, Z_density$x)
    densities = c(densities, Z_density$y)
  }

  Z_raster = data.frame(time=time, position=position, densities=densities)

  p = ggplot() +
      geom_raster(aes(x=Z_raster$time,
                      y=Z_raster$position,
                      fill=Z_raster$densities)) +
      geom_line(aes(x=1:n_iter,
                    y=sim_res$Z,
                    color="True location")) +
      geom_line(aes(x=1:n_iter,
                    y=rowMeans(particle_res$Z),
                    color="Expected location")) +
      labs(x="Time step",
           y="Location",
           fill="Particles density",
           title=title)
  return(p)
}

# Defining the ssm simulation and particle filter.
ssm_sim = function(n_iter, n_samples=1, std=1)
{
  # Observations and states.
  X = rep(-999, n_iter)
  Z = rep(-999, n_iter + 1)

  # Initiate the states.
  Z[1] = sample_init(n_samples)

  # Simulating the observations and path.
  for (i in 1:n_iter)
  {
    Z[i + 1] = sample_transition(Z[i], n=n_samples, std=std)
    X[i] = sample_emission(Z[i + 1], n=n_samples, std=std)
  }

  return(list(Z=Z[2:(n_iter + 1)], X=X, z_init=Z[1]))
}

# First method.
```

```
particle_filter_01 = function(observations, n_iter=100, n_particles=100, std=1)
{
  # Initilization of the variables.
  Z = matrix(nrow=n_iter + 1, ncol=n_particles)
  Z_bar = matrix(nrow=n_iter, ncol=n_particles)
  W = matrix(nrow=n_iter, ncol=n_particles)
  Z[1, ] = sample_init(n_particles)

  for (t in 1:n_iter)
  {
    # Prediction.
    # Transition step.
    # Prior belief.
    Z_bar[t, ] = sapply(Z[t, ], sample_transition, std=std)

    # Getting the weights.
    W[t, ] = sapply(Z_bar[t, ], density_emission, x_t=observations[t], std=std)
    W[t, ] = W[t, ] / sum(W[t, ])  # Normalizing because of computation

    # Correction.
    # Posterior belief.
    Z[t + 1, ] = sample(Z_bar[t, ],
                        size=n_particles,
                        replace=TRUE,
                        prob=W[t, ])
  }

  # Getting the means.
  EZ = rowMeans(Z[2:(n_iter + 1), ])

  return(list(Z_bar=Z_bar,
              Z=Z[2:(n_iter + 1), ],
              W=W,
              EZ=EZ))
}

# Bishops method.
particle_filter_02 = function(observations,
                              n_iter=100,
                              n_particles=100,
                              std=1,
                              correction=TRUE)
{

  # Initilization of the variables.
  Z = matrix(nrow=n_iter + 1, ncol=n_particles)
  W = matrix(1, nrow=n_iter + 1, ncol=n_particles)
  Z[1, ] = sample_init(n_particles)

  for (t in 1:n_iter)
  {
    # Z_{t+1} | X_n
    Z[t + 1, ] = sample(Z[t, ],
```

```
                        size=n_particles,
                        replace=TRUE,
                        prob=W[t, ])

    Z[t + 1, ] = sapply(Z[t + 1, ], sample_transition, std=std)

    # Update weights.
    if (correction)
    {
      W[t + 1, ] = sapply(Z[t + 1, ], density_emission, x_t=observations[t], std=std)
      W[t + 1, ] = W[t + 1, ] / sum(W[t + 1, ])
    }

  }

  # Getting the means.
  EZ = rowMeans(Z[2:(n_iter + 1), ])

  return(list(Z=Z[2:(n_iter + 1), ],
              W=W,
              EZ=EZ))

}

# Parameters.
n_iter = 100
m_particles = 100


# Simulate and use particle filter.
sim_res = ssm_sim(n_iter)
observations = sim_res$X
particle_res = particle_filter_02(observations)

# Plotting the results.
p = give_me_plot(particle_res, sim_res, "Particle Filter (Method 02)")
print(p)
```

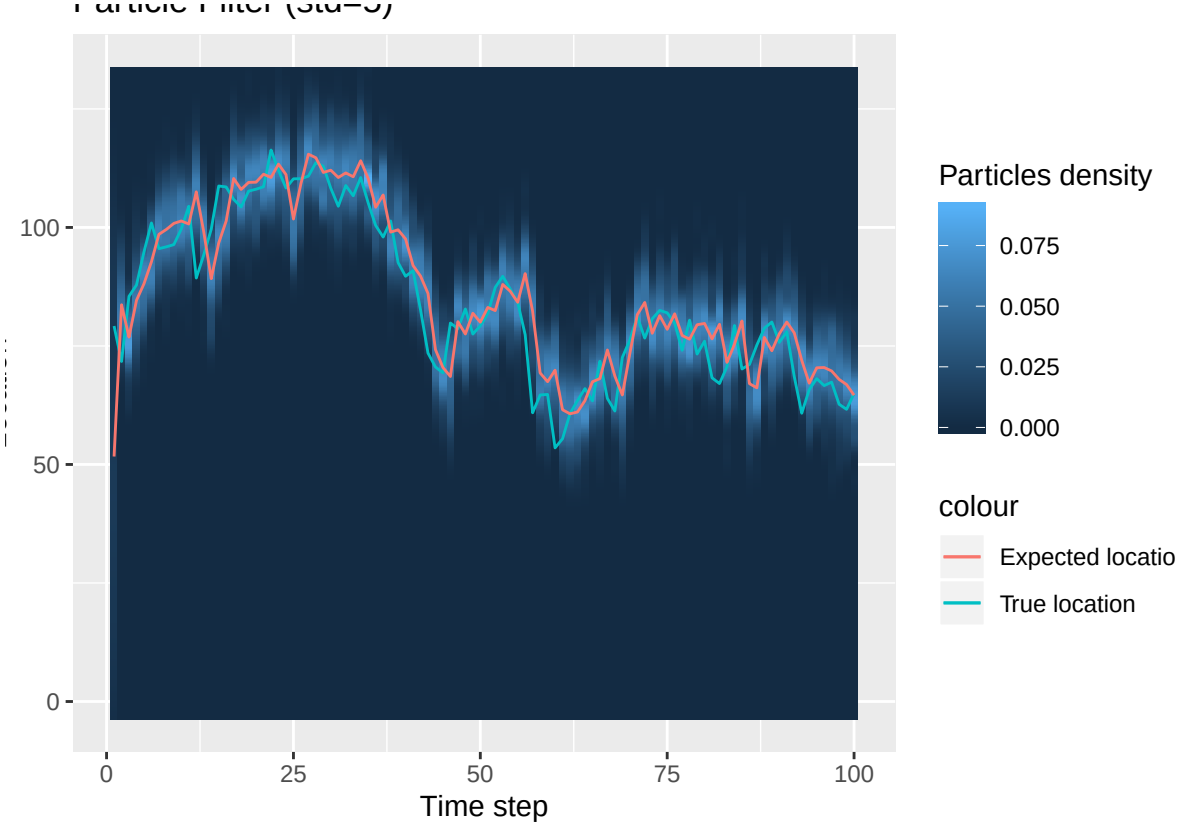


Task 2

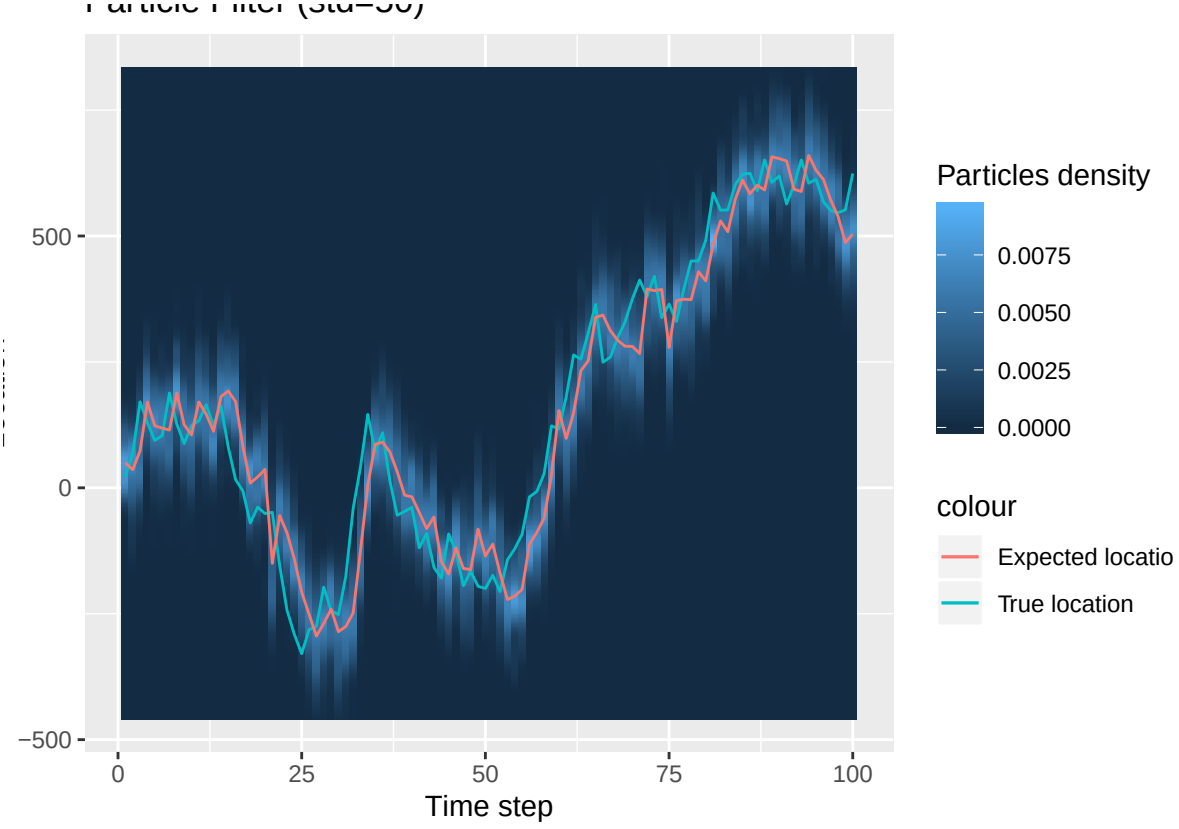
Repeat the exercise above replacing the standard deviation of the emission model with 5 and then with 50. Comment on how this affects the results.

Answer It can be seen that this affects the accuracy of the results and it introduces uncertainty into the particles distribution. This can be seen by the wider halo around the curves.

```
std = 5
sim_res = ssm_sim(n_iter, std=std)
observations = sim_res$X
particle_res = particle_filter_02(observations, std=std)
p = give_me_plot(particle_res, sim_res, "Particle Filter (std=5)")
print(p)
```




```
std = 50
sim_res = ssm_sim(n_iter, std=std)
observations = sim_res$X
particle_res = particle_filter_02(observations, std=std)
p = give_me_plot(particle_res, sim_res, "Particle Filter (std=50)")
print(p)
```

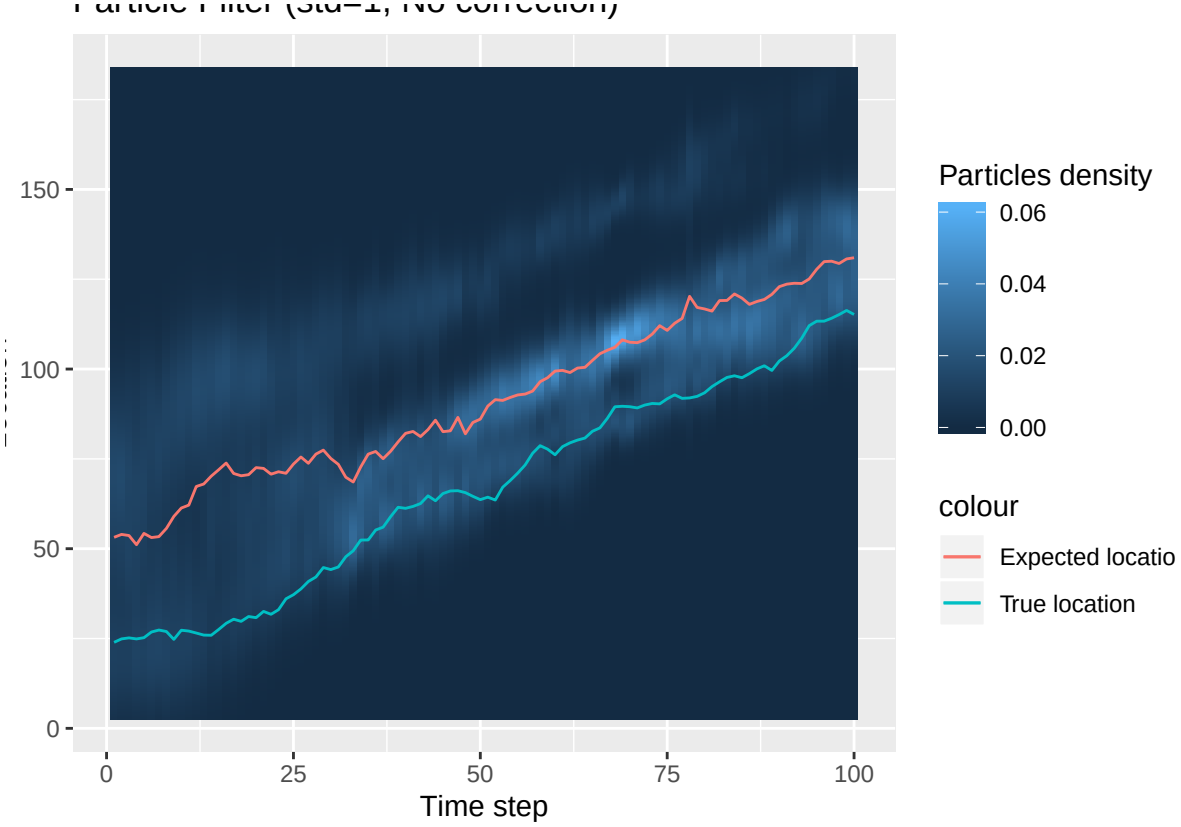


Task 3

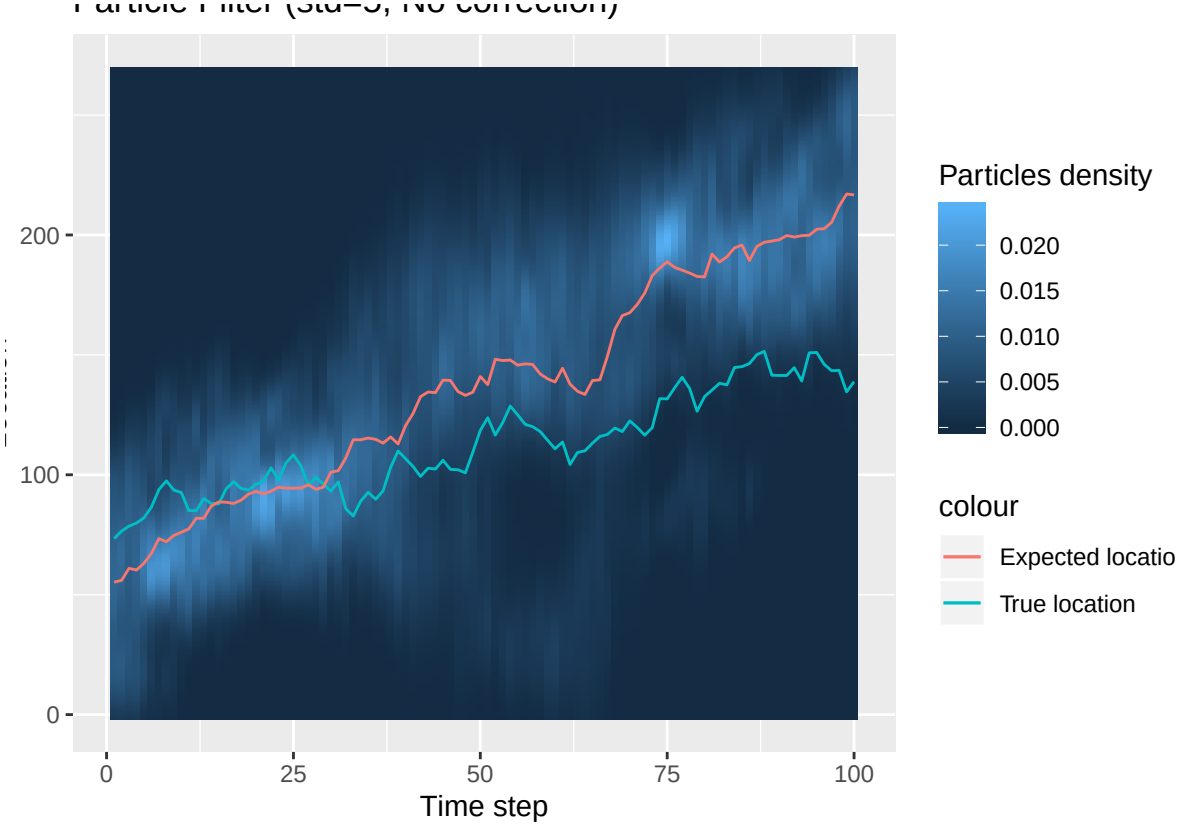
Finally, show and explain what happens when the weights in the particle filter are always equal to 1, i.e. there is no correction.

Answer If there is no correction we see how the particles sample doesn't incorporate any of the observations and thus our particle filter transforms itself into a time series simulator where the next time step is defined by the transition model.

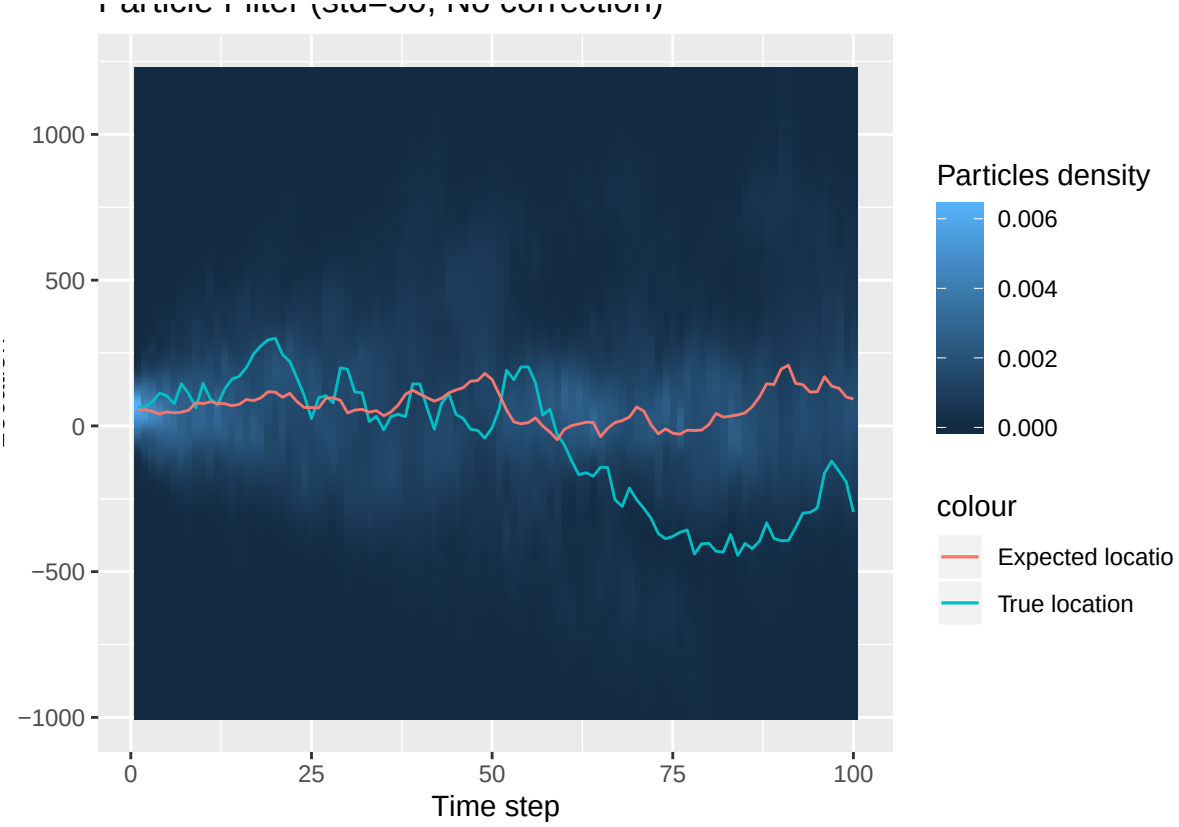
```
std = 1
sim_res = ssm_sim(n_iter, std=std)
observations = sim_res$X
particle_res = particle_filter_02(observations, std=std, correction=FALSE)
p = give_me_plot(particle_res, sim_res, "Particle Filter (std=1, No correction)")
print(p)
```




```
std = 5
sim_res = ssm_sim(n_iter, std=std)
observations = sim_res$X
particle_res = particle_filter_02(observations, std=std, correction=FALSE)
p = give_me_plot(particle_res, sim_res, "Particle Filter (std=5, No correction)")
print(p)
```




```
std = 50
sim_res = ssm_sim(n_iter, std=std)
observations = sim_res$X
particle_res = particle_filter_02(observations, std=std, correction=FALSE)
p = give_me_plot(particle_res, sim_res, "Particle Filter (std=50, No correction)")
print(p)
```


1000
500
0
−500
−1000
0
25
50
75
100
Time step
Particles density
0.006
0.004
0.002
0.000
colour
Expected locatio
True location