Advanced Machine Learning - lab03

Lennart Schilling (lensc874)

2019-10-07

Contents

Lab03. State Space Models.

Assignment 1. Localizing a robot using the particle filter.

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:

$$p(z_t|z_{t-1}) = (\mathcal{N}(z_t|z_{t-1},1) + \mathcal{N}(z_t|z_{t-1}+1,1) + \mathcal{N}(z_t|z_{t-1}+2,1))/3 \quad \text{// Transition model}$$

$$p(x_t|z_t) = (\mathcal{N}(x_t|z_t,1) + \mathcal{N}(x_t|z_t-1,1) + \mathcal{N}(x_t|z_t+1,1))/3 \quad \text{// Emission model}$$

$$p(z_1) = Uniform(0,100) \quad \text{// Initial model}$$

1a. Implementing SSM. Simulating data. Running particle filter. Comparing results.

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.

Defining functions for initial, emission and transition model.

```
# Defining function to sample from intial model.
r_initial = runif(n = 1, min = 0, max = 100)
# Defining function to sample from emission model.
r_emission = function(state, sd) {
  plus = sample(c(0,-1,1), 1)
  return(rnorm(n = 1, mean = state + plus, sd = sd))
}
# Defining function to sample from transition model.
r_transition = function(state_previous, sd) {
  plus =  sample(c(0,1,2), 1)
  return(rnorm(n = 1, mean = state_previous + plus, sd = sd))
}
# Defining function to compute density of emission model for a given observation.
d_emission = function(obs, state, sd) {
  return(sum(dnorm(x = obs, mean = state, sd = sd),
             dnorm(x = obs, mean = state - 1, sd = sd),
             dnorm(x = obs, mean = state + 1))/3)
}
```

Simulating states & observations from given State Space model.

```
# Implementing general function for simulation of states and observations.
generate_from_true_ssm = function(T, # Maximum time steps.
                                  f_initialization, # Function for initialization.
                                  f_sample_emission, # Function to sample from emission model.
                                  sd_emission, # sd of emission model.
                                  f_sample_transition, # Function to sample from transition model.
                                  sd_transition) { # sd of transition model.

  # Creating vectors to store sampled states and observations.
  states = c()
  obs = c()
  # Inizializing state and observation for t_1.
```

```
    # State.
    states[1] = f_initialization
    # Observation.
    obs[1] = f_sample_emission(state = states[1], sd = sd_emission)
  # Sampling states and observations for t = 2, ..., T.
  for (t in (2:T)) {
    # State.
    states[t] = f_sample_transition(state_previous = states[t-1], sd = sd_transition)
    # Observation.
    obs[t] = f_sample_emission(state = states[t], sd = sd_emission)
  }
  # Returning sampled values.
  return(list(states = states, obs = obs))
}

# Simulating states and observations from given SSM.
sample_true_smm = generate_from_true_ssm(
  T = 100,
  f_initialization = r_initial,
  f_sample_emission = r_emission,
  sd_emission = 1,
  f_sample_transition = r_transition,
  sd_transition = 1)
```

Identifying hidden states via particle filtering.

```
# Implementing general function to run particle filter.
particle_filter = function(M, # Number of particles.
                           obs, # Observations.
                           f_initialization, # Function for initialization.
                           f_sample_transition, # Function to sample from emission model.
                           sd_transition, # sd of transition model.
                           f_dens_emission, # Function to get density from emission model.
                           sd_emission, # sd of emission model.
                           correction = TRUE) { # Computing weights and peforming correction?

  # Extracting T from given observations.
  T = length(obs)
  # Creating matrix to store all particles for all t.
  particles = matrix(data = NA,
                     nrow = T, # Times 1:T.
                     ncol = M, # Particles 1:M.
                     dimnames = list(paste0("t_",c(1:T)), paste0("m_", c(1:M))))
  # Creating vector to store temporary, non-corrected, predicted particles.
  particles_temp = c()
  # Creating vector to store weights (emission densities) of sampled particles.
  weights = c()
  # INITIALIZATION.
  particles[1, ] = f_initialization
  for (t in 2:T) { # For each time.
    for (m in 1:M) { # For each particle.
      # PREDICTION.
      particles_temp[m] = f_sample_transition(state_previous = particles[t-1, m], sd = sd_transition)
      # GETTING IMPORTANCE WEIGHT.
```

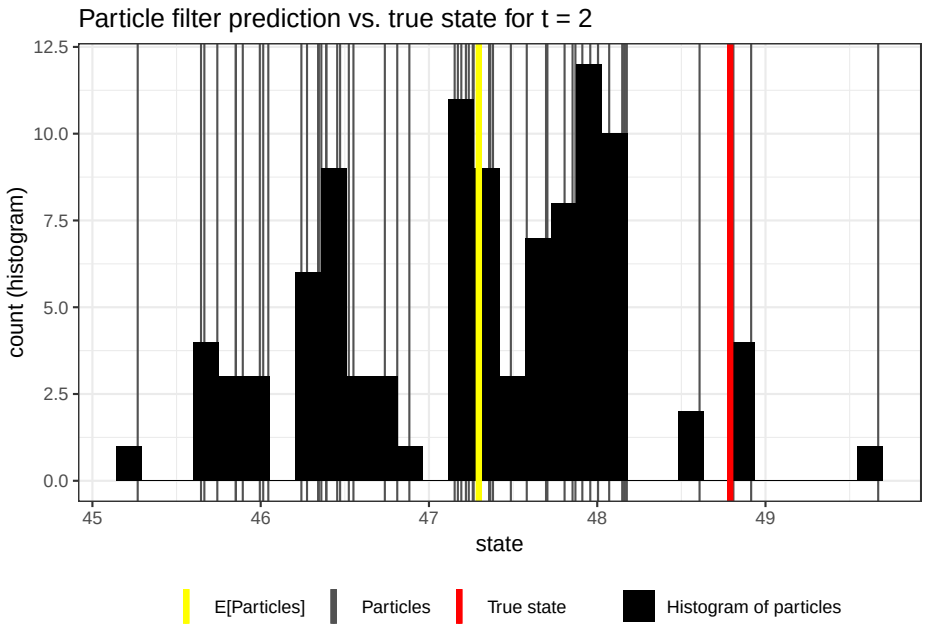
```
      if (correction) {
        weights[m] = f_dens_emission(obs = obs[t], state = particles_temp[m], sd = sd_emission)
      }
    }
    # CORRECTION.
    if (correction) {
      particles[t, ] = sample(x = particles_temp, # Predicted particles.
                              size = M, # M particles should be chosen.
                              replace = T,
                              prob = weights) # Particles are chosen according to the weight.
    } else {
      particles[t, ] = particles_temp
    }
  }
  # Returning all particles.
  return(particles)
}

# Running particle filter.
particles = particle_filter(M = 100,
                            obs = sample_true_smm$obs,
                            f_initialization = r_initial,
                            f_sample_transition = r_transition,
                            sd_transition = 1,
                            f_dens_emission = d_emission,
                            sd_emission = 1)
```

Visualizing results.

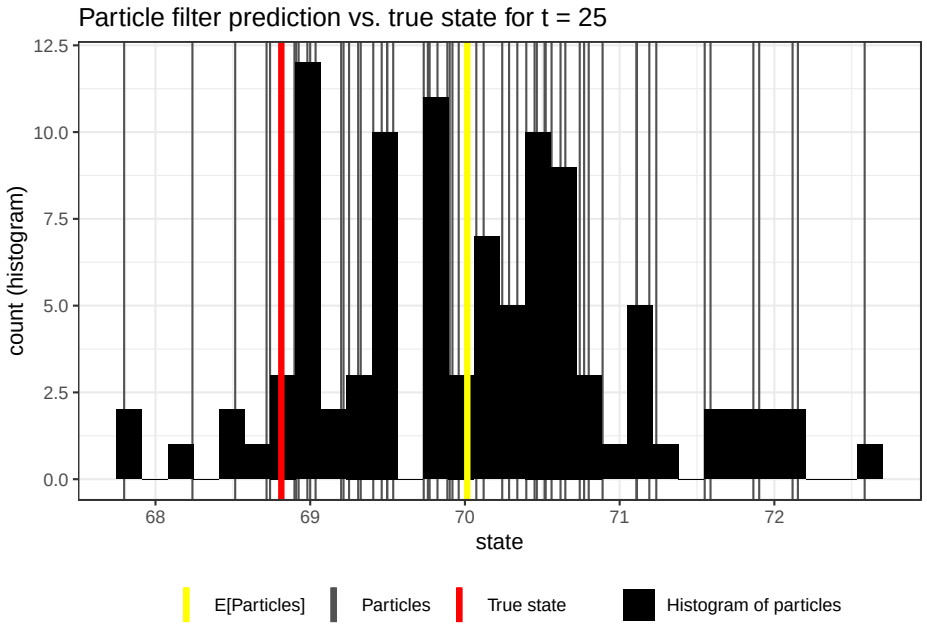
Since $t = 1$ represents our initialization time step, we will visualize for $t = 2$, $t = 25$, $t = 75$ and $t = 100$.

```
library(ggplot2)
# Implementing function for visualization.
visualize_particles = function(t, # time step.
                               obs, # observations
                               true_states,
                               particles_matrix) {
  # Initializing plot.
  p = ggplot()
  p = p +
    # Adding all particles.
    geom_vline(aes(xintercept = particles_matrix[t, ],
                   color = "Particles")) +
    # Adding histogram of particles.
    geom_histogram(aes(x = particles_matrix[t,],
                       fill = "Histogram of particles")) +
    # Adding Expected value of particles.
    geom_vline(aes(xintercept = mean(particles_matrix[t, ]),
                   color = "E[Particles]"),
               size = 1.5) +
    # Adding true state.
    geom_vline(aes(xintercept = true_states[t],
                   color = "True state"),
               size = 1.5) +
```

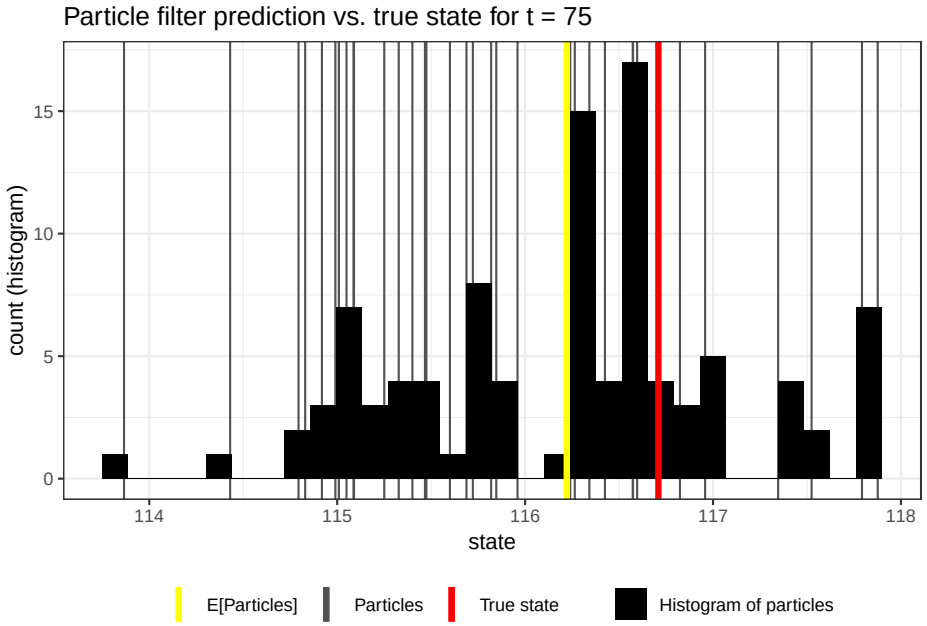
```
    # Setting up layout.
    scale_color_manual(values = c("True state" = "red",
                                  "Particles" = "grey32",
                                  "E[Particles]" = "yellow")) +
    scale_fill_manual(values = c("Histogram of particles" = "black")) +
    theme_bw() +
    theme(legend.position="bottom") +
    labs(title = paste0("Particle filter prediction vs. true state for t = ", t),
         x = "state",
         y = "count (histogram)",
         color = NULL,
         fill = NULL)
  # Returning plot.
  return(p)
}

# Running visualization function for t = 2, 25, 75, 100.
visualize_particles(t = 2,
                    obs = sample_true_smm$obs,
                    true_states = sample_true_smm$states,
                    particles_matrix = particles)
```



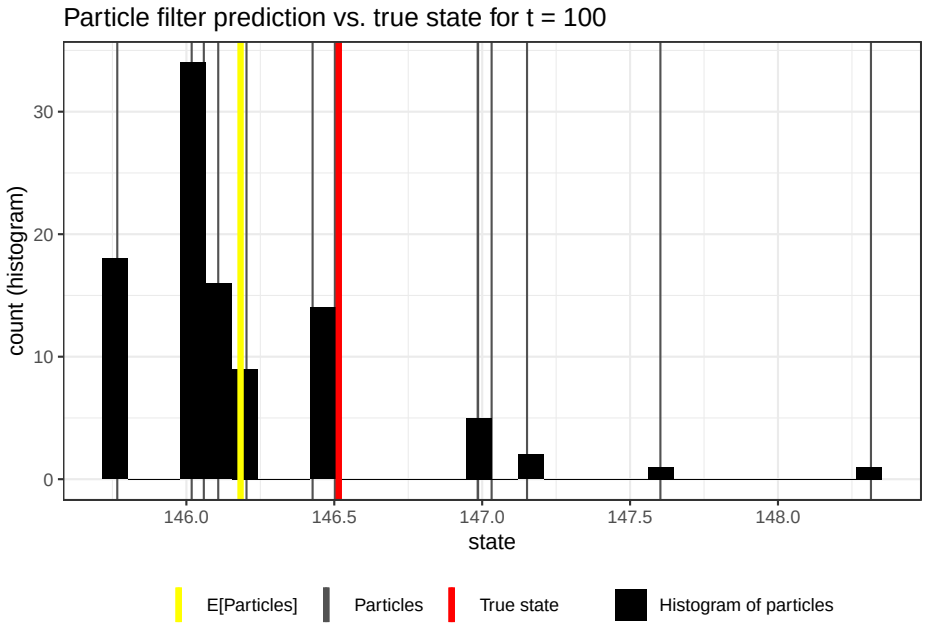

```
visualize_particles(t = 25,
                    obs = sample_true_smm$obs,
                    true_states = sample_true_smm$states,
                    particles_matrix = particles)
```




```
visualize_particles(t = 75,
                    obs = sample_true_smm$obs,
                    true_states = sample_true_smm$states,
                    particles_matrix = particles)
```




```
visualize_particles(t = 100,
                    obs = sample_true_smm$obs,
                    true_states = sample_true_smm$states,
                    particles_matrix = particles)
```

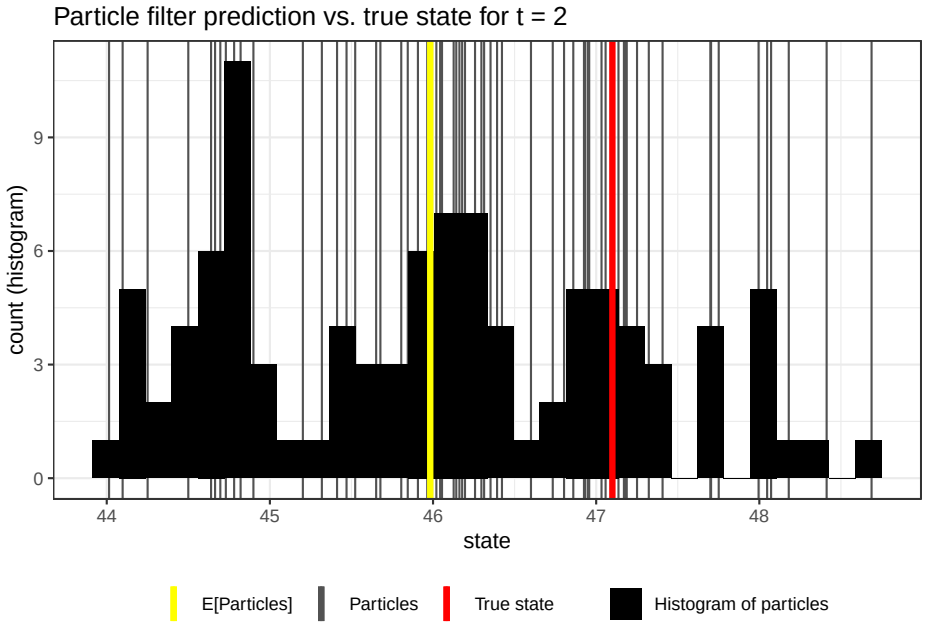

Particle filter prediction vs. true state for t = 100
count (histogram)
30
20
10
0
146.0
146.5
147.0
147.5
148.0
state
E[Particles]
Particles
True state
Histogram of particles

1b. Repeating process with adjusted standard deviations of emission model.

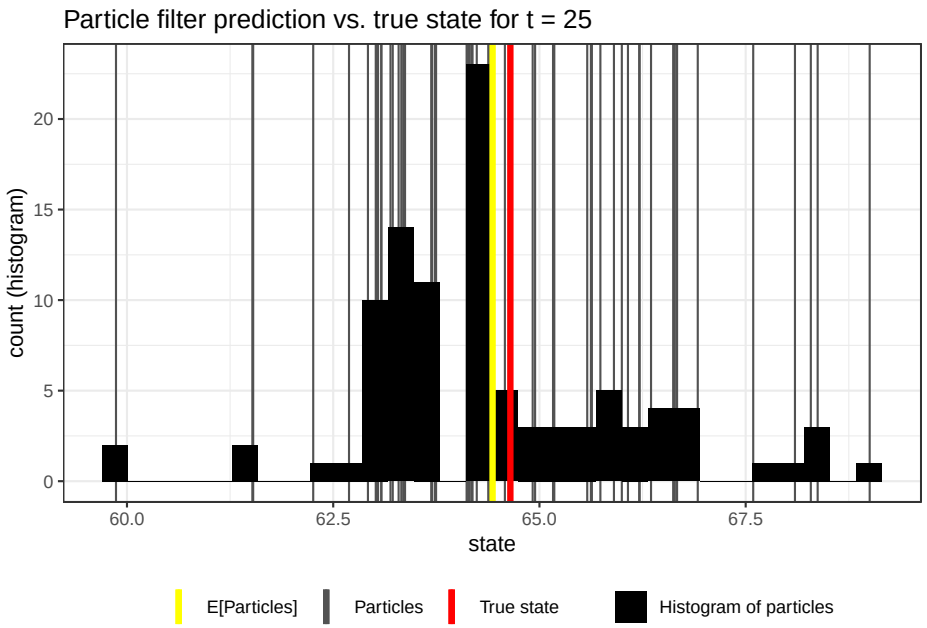
Repeat the exercise above replacing the standard deviation of the emission model with 5 and then with 50. Comment on how this affects the results.

Replacing standard deviation of emission model by 5.

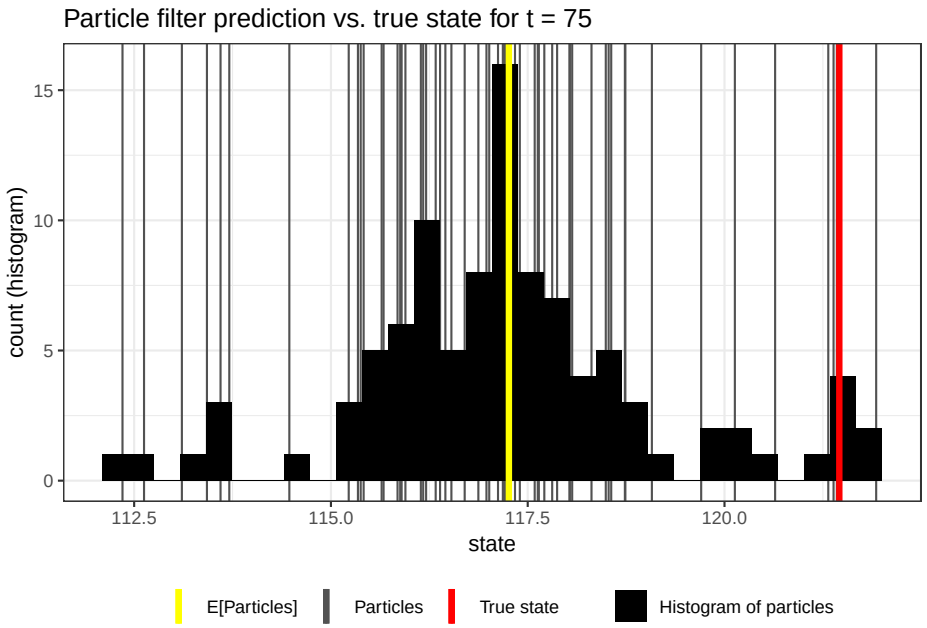
```
# Simulating states and observations from given SSM.
sample_true_smm = generate_from_true_ssm(T = 100,
                                         f_initialization = r_initial,
                                         f_sample_emission = r_emission,
                                         sd_emission = 5, # Replaced by 5.
                                         f_sample_transition = r_transition,
                                         sd_transition = 1)

# Running particle filter.
particles = particle_filter(M = 100,
                            obs = sample_true_smm$obs,
                            f_initialization = r_initial,
                            f_sample_transition = r_transition,
                            sd_transition = 1,
                            f_dens_emission = d_emission,
                            sd_emission = 5) # Replaced by 5.

# Visualizing for t = 2, 25, 75, 100.
visualize_particles(t = 2,
                    obs = sample_true_smm$obs,
                    true_states = sample_true_smm$states,
                    particles_matrix = particles)
```



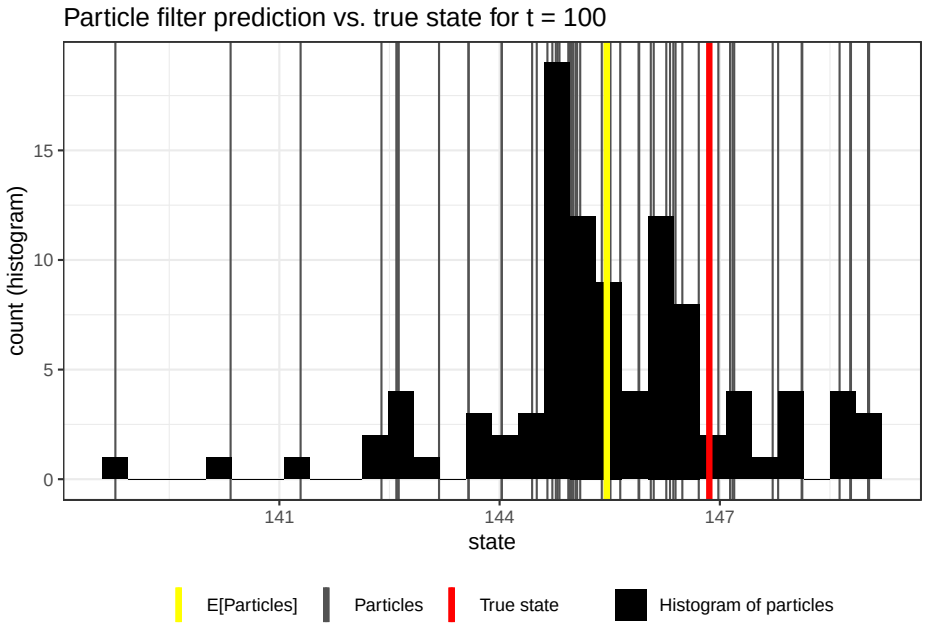

```
visualize_particles(t = 25,
                    obs = sample_true_smm$obs,
                    true_states = sample_true_smm$states,
                    particles_matrix = particles)
```




```
visualize_particles(t = 75,
                    obs = sample_true_smm$obs,
                    true_states = sample_true_smm$states,
                    particles_matrix = particles)
```




```
visualize_particles(t = 100,
                    obs = sample_true_smm$obs,
                    true_states = sample_true_smm$states,
                    particles_matrix = particles)
```

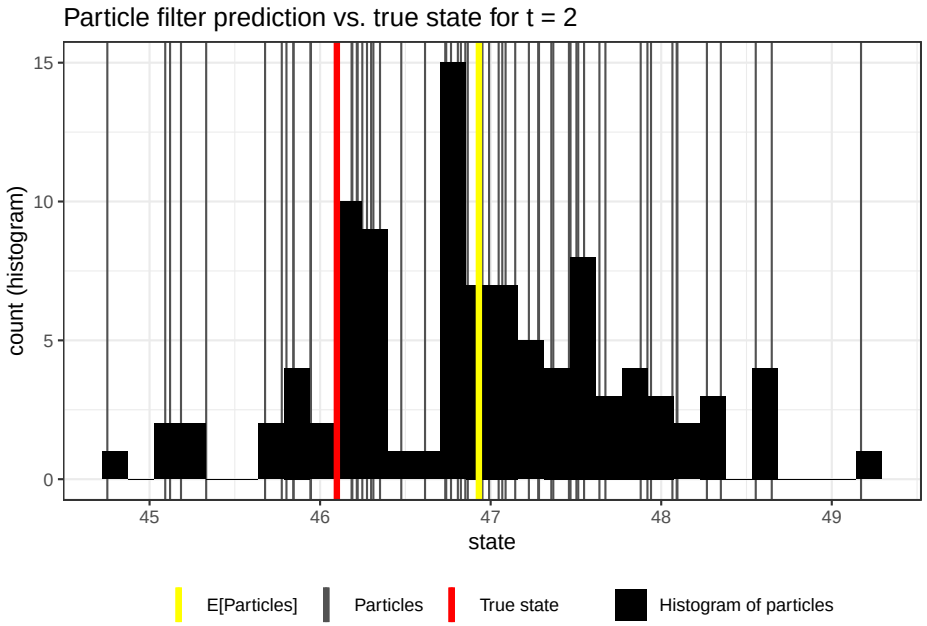


Conclusions:

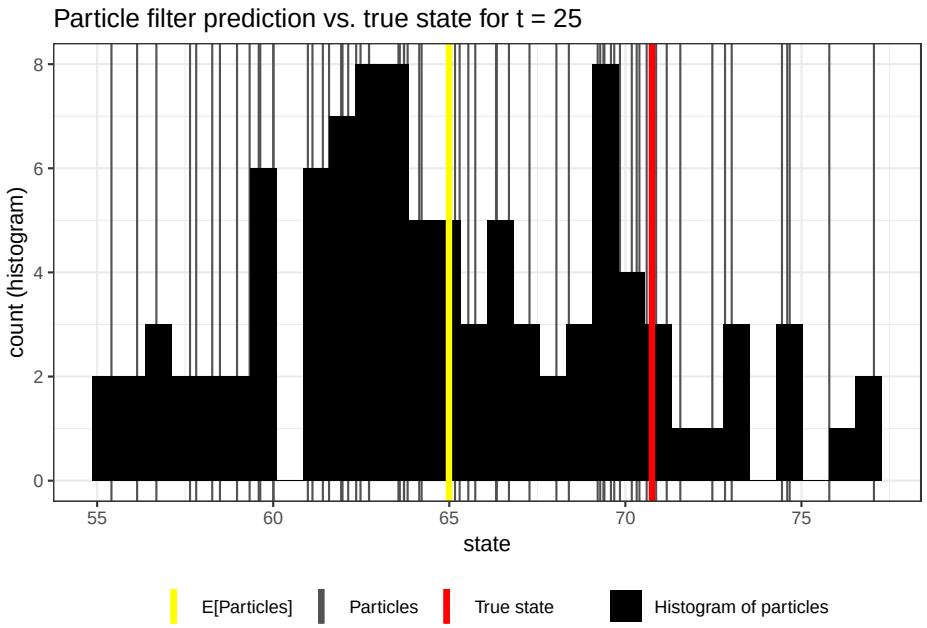
Analyzing these plots, one should pay attention to the scale of the x-axis. The easiest way to assess the accuracy of the plot is probably to compare the expected state of the particles to the actual true state. It becomes obvious that a higher time step `t` increases the chance of a larger difference between the "guess" and the true state. Also, compared to the plots from assignment *1a* (with standard deviation of emission model equal to 1), the chance of having a larger difference seems to be higher.

Replacing standard deviation of emission model by 50.

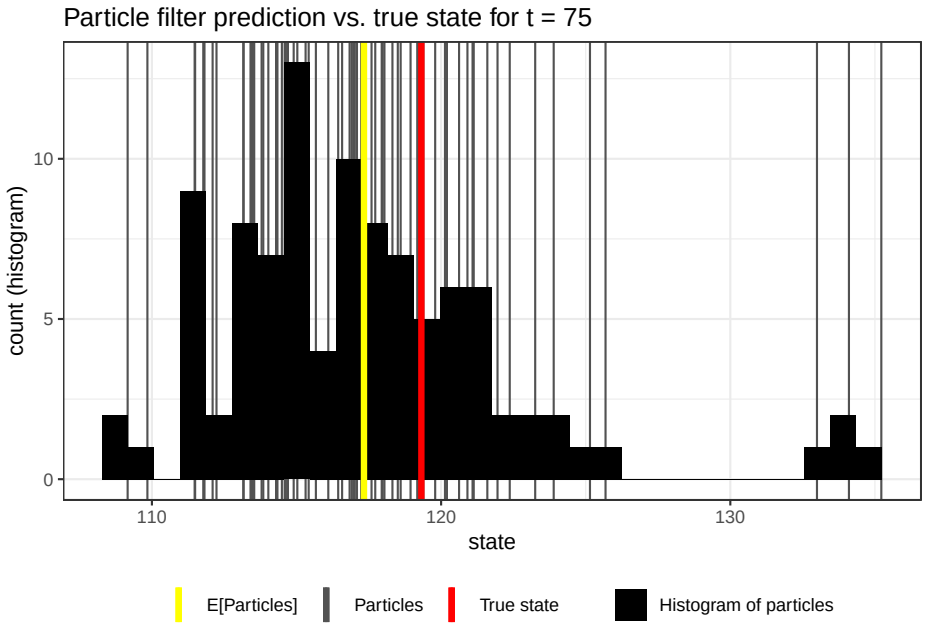
```
# Simulating states and observations from given SSM.
sample_true_smm = generate_from_true_ssm(T = 100,
                                         f_initialization = r_initial,
                                         f_sample_emission = r_emission,
                                         sd_emission = 50, # Replaced by 50.
                                         f_sample_transition = r_transition,
                                         sd_transition = 1)

# Running particle filter.
particles = particle_filter(M = 100,
                            obs = sample_true_smm$obs,
                            f_initialization = r_initial,
                            f_sample_transition = r_transition,
                            sd_transition = 1,
                            f_dens_emission = d_emission,
                            sd_emission = 50) # Replaced by 50.

# Visualizing for t = 2, 25, 75, 100.
visualize_particles(t = 2,
                    obs = sample_true_smm$obs,
                    true_states = sample_true_smm$states,
                    particles_matrix = particles)
```



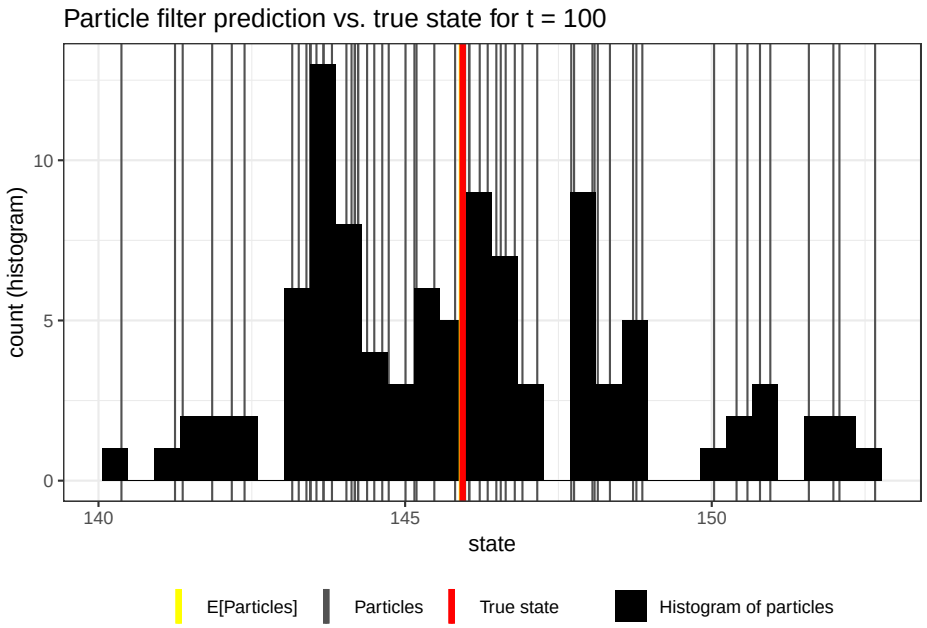

```
visualize_particles(t = 25,
                    obs = sample_true_smm$obs,
                    true_states = sample_true_smm$states,
                    particles_matrix = particles)
```




```
visualize_particles(t = 75,
                    obs = sample_true_smm$obs,
                    true_states = sample_true_smm$states,
                    particles_matrix = particles)
```




```
visualize_particles(t = 100,
                    obs = sample_true_smm$obs,
                    true_states = sample_true_smm$states,
                    particles_matrix = particles)
```



Conclusions:

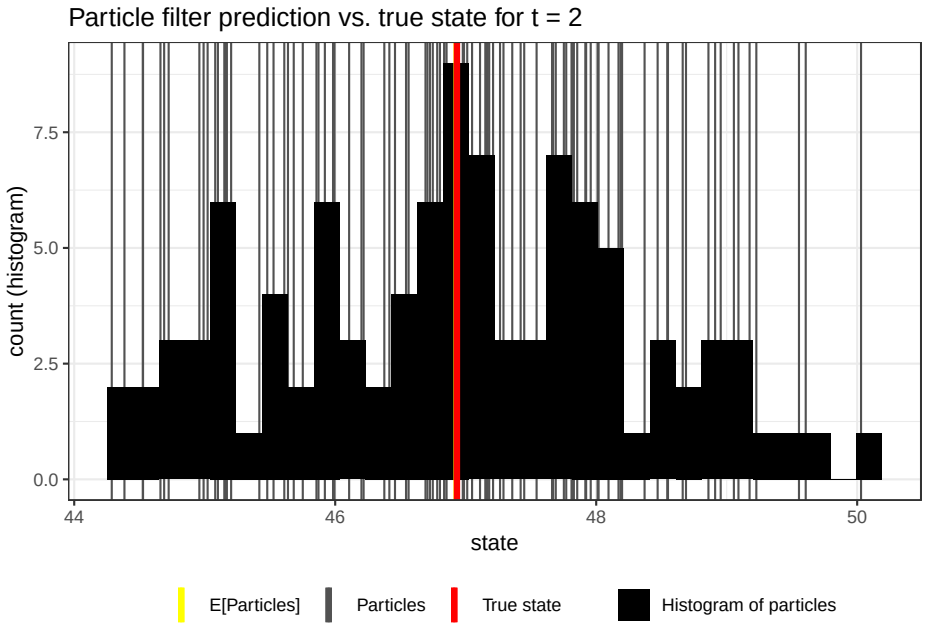
Again, considering the scale of the x-axis, one can see that at first, generally the difference between the particles and the actual state seems to rise over time (difference is larger for a higher t). Also, compared to the previous run (standard deviation of emission model = 5), this chosen standard deviation (50) leads to a much larger difference between the particles and the actual state. This can be explained by the fact that the model expcts a higher observation error. Thus, the particle filter algorithm takes this into account within the *correction*-step.

1c. Repeating process without correcting the particle filter predictions.

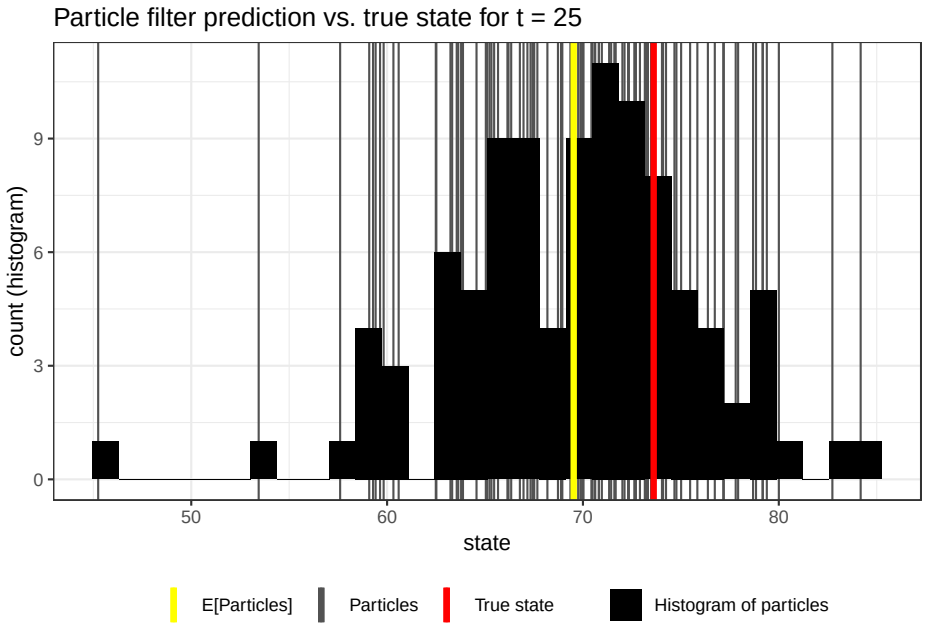
Finally, show and explain what happens when the weights in the particle filter are always equal to 1, i.e. there is no correction.

For this assignment, we can set the implemented paramter `correction` within the function `particle_filter()` to `FALSE`. This prevents the correction so that the non-corrected predictions will be returned by the algorithm. To show the effects, we will again repeat the same process from before (standard deviation of the emission matrix will be set to 1 again).

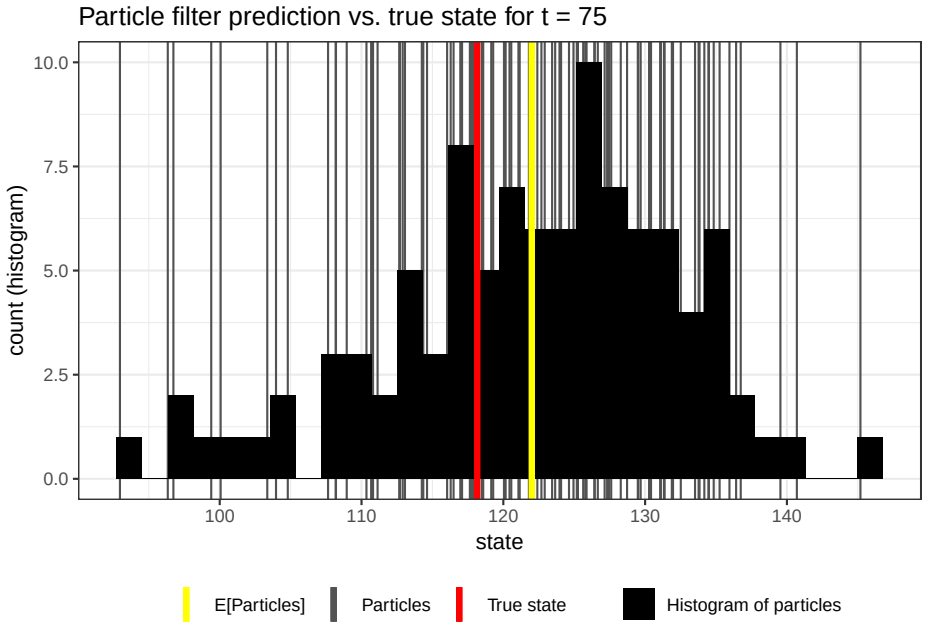
```
# Simulating states and observations from given SSM.
sample_true_smm = generate_from_true_ssm(T = 100,
                                         f_initialization = r_initial,
                                         f_sample_emission = r_emission,
                                         sd_emission = 1,
                                         f_sample_transition = r_transition,
                                         sd_transition = 1)

# Running particle filter.
particles = particle_filter(M = 100,
                            obs = sample_true_smm$obs,
                            f_initialization = r_initial,
                            f_sample_transition = r_transition,
                            sd_transition = 1,
                            f_dens_emission = d_emission,
                            sd_emission = 1,
                            correction = FALSE) # Preventing correction.

# Visualizing for t = 2, 25, 75, 100.
visualize_particles(t = 2,
                    obs = sample_true_smm$obs,
                    true_states = sample_true_smm$states,
                    particles_matrix = particles)
```



```
visualize_particles(t = 25,
                    obs = sample_true_smm$obs,
                    true_states = sample_true_smm$states,
                    particles_matrix = particles)
```

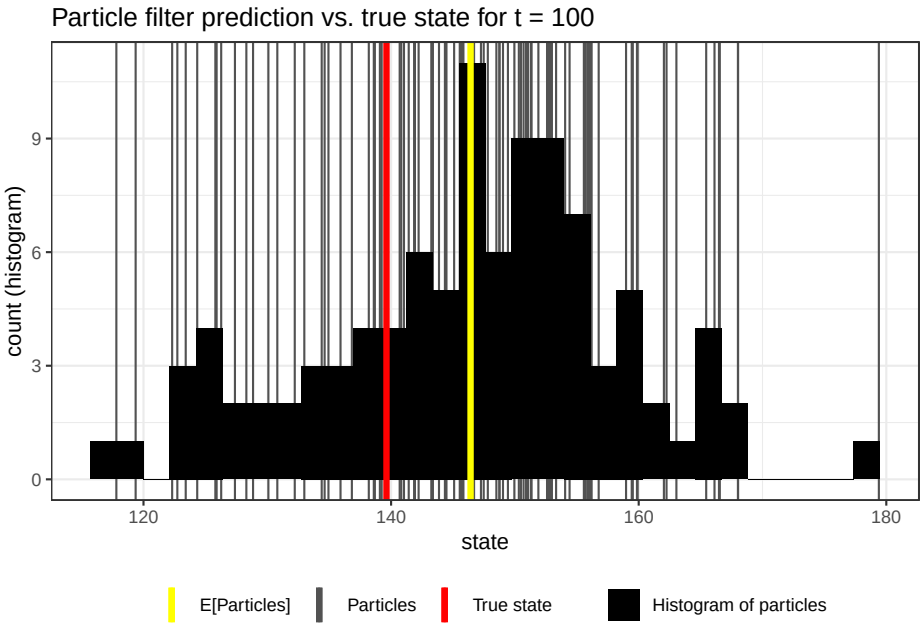



```
visualize_particles(t = 75,
                    obs = sample_true_smm$obs,
                    true_states = sample_true_smm$states,
                    particles_matrix = particles)
```




```
visualize_particles(t = 100,
                    obs = sample_true_smm$obs,
```

```
                    true_states = sample_true_smm$states,
                    particles_matrix = particles)
```



Conclusions:

Even if the plots look quite similar to the previous ones, considering the scale of the x-axis, one can see that the differences are indeed larger than before. Some plots show a very large difference between the expected value of the particles and the true state. Thus, as expected, the identified hidden states (in form of particles) are not that accurate anymore related to the actual true hidden states.