732A96: Advanced Machine Learning

Computer Lab 3: State Space Models

Hariprasath Govindarajan (hargo729)

October 3, 2019

Contents

NOTATIONS: The hidden states are denoted as z_t and the observations are denoted as x_t.

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:

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

Emission Model:

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

Initial Model:

$$p(z_1) = Uniform(0, 100)$$

Question 1: SSM simulation and Particle filter implementation

Question: 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.

Answer:

SSM Simulation

We create the function `simSSM()` which can be used to simulate from a SSM and it requires the following parameters:

- `N` : Number of time steps to simulate
- `initial_model` : A function which generates the state for the first time step (z_1)
- `transition_model` : A function which generates the state, z_t given z_{t-1}
- `emission_model`: A function that generates the observation, x_t given the state, z_t

The function returns a list containing the components - `states` and `observations`.

```r
# Function to simulate states and observations for a State Space Model
# using the given initial model, transition model and emission model
simSSM = function(N, initial_model, transition_model, emission_model){
  z_t = vector(mode = "numeric", length = N)

  # Simulate initial state
  z_t[1] = initial_model()

  # Simulate remaining states
  for(i in 2:N){
    z_t[i] = transition_model(z_t[i-1])
  }

  # Simulate observations from states
  x_t = sapply(z_t, emission_model)

  res = list(states = z_t, observations = x_t)

  return(res)
}
```

We create the initial, transition and emission models based on the given probability models. Then, we simulate 100 time steps using this SSM.

```r
# Transition model - mixture of 3 gaussians
transition_model = function(z_t_1){
  mean_z_t = sample(c(z_t_1, z_t_1+1, z_t_1+2), size = 1, prob = rep(1/3, 3))
  return(rnorm(1, mean_z_t, 1))
}

# Emission model - mixture of 3 gaussians (SD = 1)
emission_model = function(z_t, sd = 1){
  mean_x_t = sample(c(z_t, z_t+1, z_t-1), size = 1, prob = rep(1/3, 3))
  return(rnorm(1, mean_x_t, sd))
}

# Initial model - Unif(0, 100)
initial_model = function(){
  return(runif(1, min = 0, max = 100))
}

# Probability function for emission model
emission_prob = function(x_t, z_t, sd = 1){
  return((dnorm(x_t, mean = z_t, sd = sd) +
            dnorm(x_t, mean = z_t-1, sd = sd) +
            dnorm(x_t, mean = z_t+1, sd = sd)) / 3)
}

# Simulate 100 time steps of SSM
set.seed(12345)
ssm_sim = simSSM(100, initial_model, transition_model, emission_model)
ssm_obs = ssm_sim$observations
ssm_states = ssm_sim$states
```

Particle Filter

To perform the particle filtering to identify the states, we create the function `particle_filter()` which requires the following parameters:

- `num_particles` : Number of particles to use (M)
- `observations` : Observations from SSM (x_t)
- `initial_model` : A function which generates the state for the first time step ($z_1 \sim p(z_1)$)
- `transition_model` : A function which generates the next state given a previous state ($z_t \sim p(z_t|z_{t-1})$)
- `emission_model` : A function that generates the observation corresponding to a state ($x_t \sim p(x_t|z_t)$)

- `emission_prob` : A function that returns the probability of the observation given the state ($p(x_t|z_t)$)
- `method` : Method-1 samples from the belief set of particles ($\bar{\chi}_t$) using weighted probabilities (w_t). Method 2 samples particles from the mixture model where the transition model is weighted using the probabilities, $z_t^m \sim \sum_{m=1}^{M} w_t^m p(z_{t+1}|z_t^m)$. (default = 2)
- `weight_update`: Boolean, whether to update the weights, w_t in each time step (default = `TRUE`)

```
# Function to perform particle filtering
particle_filter = function(num_particles, observations, initial_model,
                           transition_model, emission_model, emission_prob,
                           method = 2, weight_update = TRUE){
  # Number of time steps
  N = length(observations)

  # Matrix to store the particles at all time steps
  particle_history = matrix(nrow = N, ncol = num_particles)

  # Generate initial particles from initial model
  particles = sapply(1:num_particles, function(i) initial_model())

  # Weights
  w_t = rep(1, num_particles)

  for(t in 1:N){
    # Method 1
    if(method == 1){
      # Vector for belief set of particles
      bel_particles = vector(mode = "numeric", length = num_particles)

      for(j in 1:num_particles){
        # Generate belief particle using transition model
        # z_t^m given z_t-1
        bel_particles[j] = transition_model(particles[j])

        if(weight_update){
          # Calculate weight using emission model probability
          # probability of x_t given z_t^m
          w_t[j] = emission_prob(observations[t], bel_particles[j])
        }
      }

      # Normalize weights
      w_t = w_t/sum(w_t)

      # Resample particles from belief set of particles
      # using weighted probabilities with replacement
```

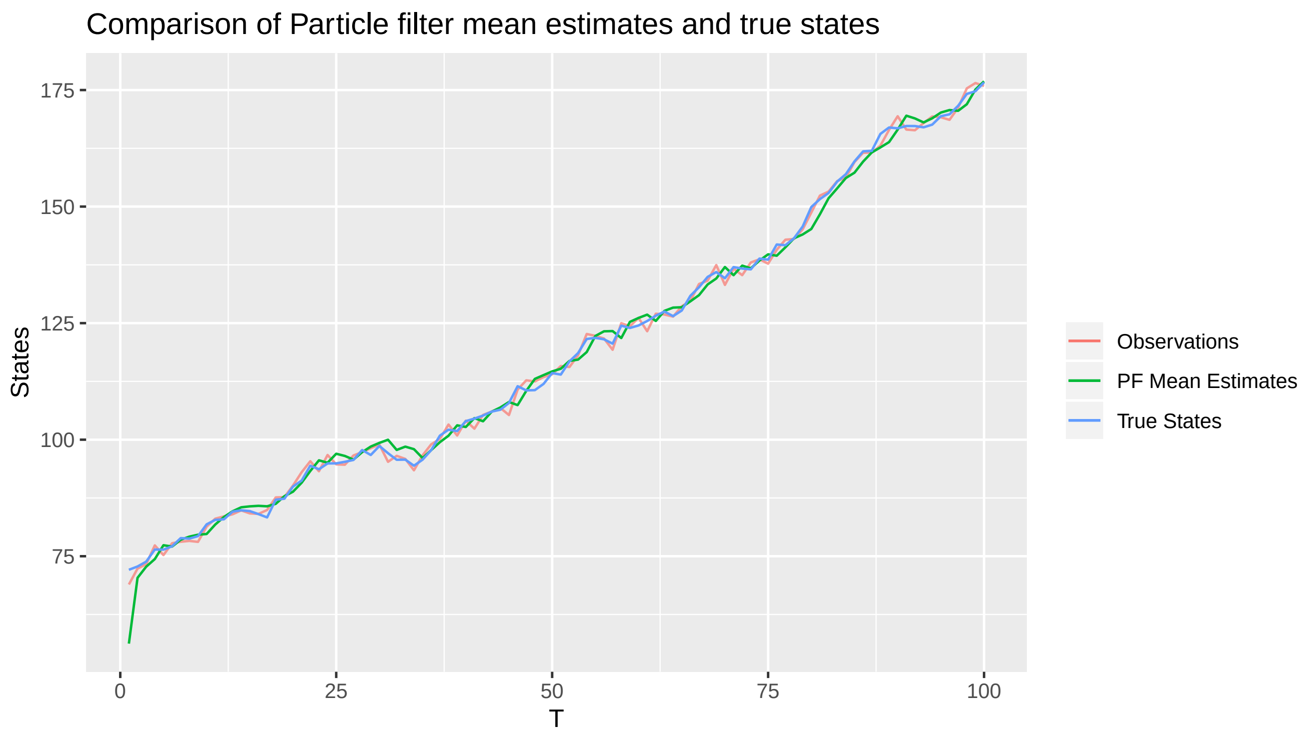
```r
      particles = sample(bel_particles, size = num_particles,
                         prob = w_t, replace = TRUE)
    }
    # Method 2
    else{
      # Sample particles from the mixture model of weighted transition models

      # Sample component mean from particles at time t-1
      # using weighted probabilities with replacement
      # z_(t-1)^m given x_t-1
      post_sample = sample(particles, size = num_particles,
                           prob = w_t, replace = TRUE)

      for(j in 1:num_particles){
        # Resample particles based on transition model and sampled component mean
        # z_t given z_t-1
        particles[j] = transition_model(post_sample[j])

        if(weight_update){
          # Calculate weight using emission model probability
          # probability of x_t given z_t
          w_t[j] = emission_prob(observations[t], particles[j])
        }
      }

      # Normalize weights
      w_t = w_t/sum(w_t)
    }

    # Add the set of particles to particle history
    particle_history[t, ] = particles
  }

  # Calculate mean estimates of particles for each time step
  mean_estimates = rowMeans(particle_history)

  res = list(pf_history = particle_history,
             pf_means = mean_estimates)

  return(res)
}
```

Now, we perform particle filtering using 100 particles based on the observations that we simulated earlier. We also calculate the residuals as the difference between the mean estimates obtained from the particles at each time step and the true states from the simulation.

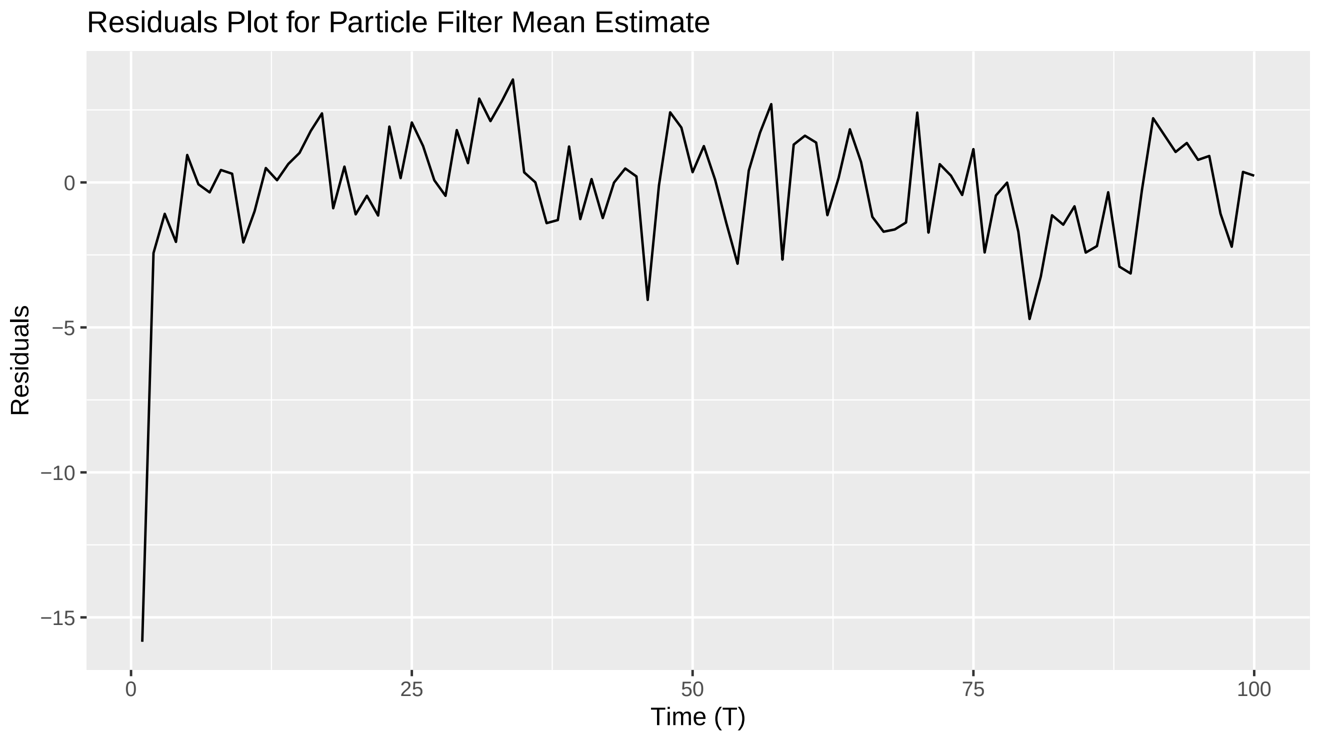
```r
# Running particle filter
pf_res = particle_filter(100, ssm_obs, initial_model, transition_model,
                         emission_model, emission_prob, method = 2)
residuals = pf_res$pf_means - ssm_states
```

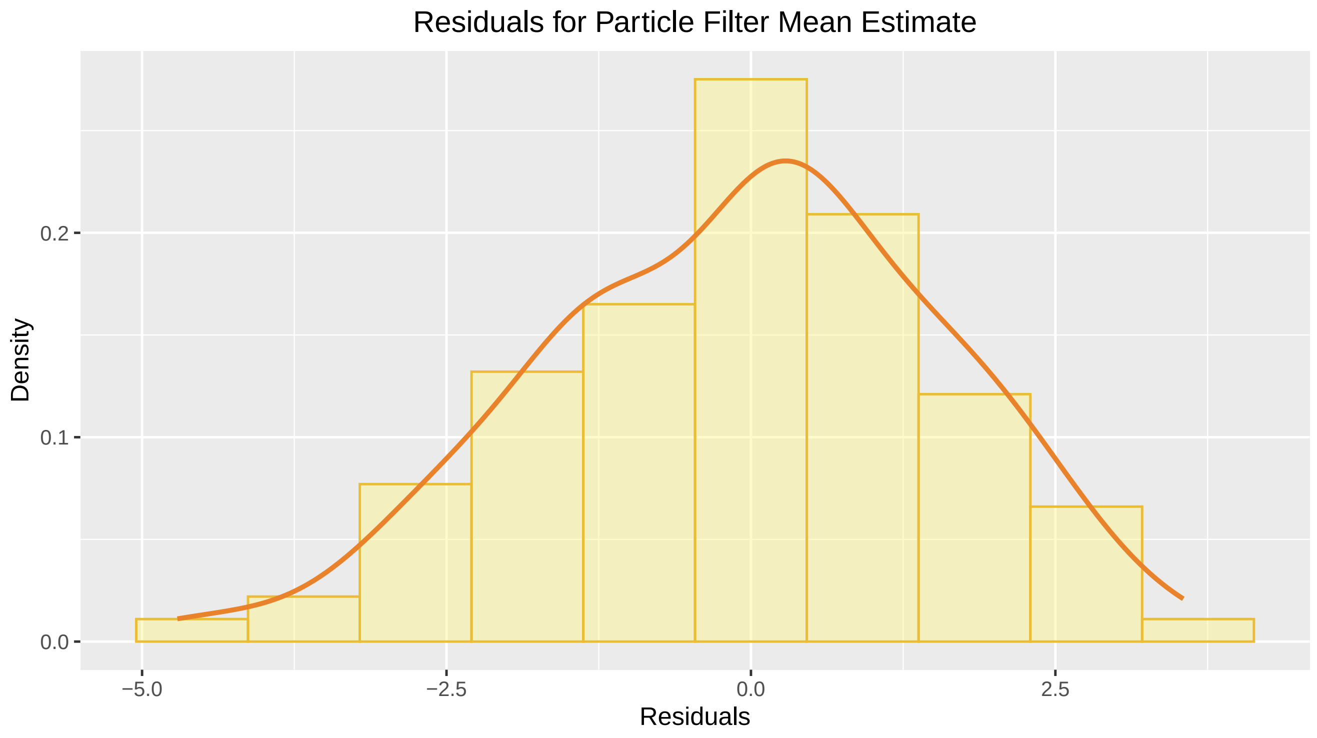
Analysis of particle filter results

The following plot shows the mean estimates of the particles, the true states (z_t) and the observations(x_t) at each time step.



The residual plot and the distribution of the residuals based on the mean estimates of the particles are as follows.

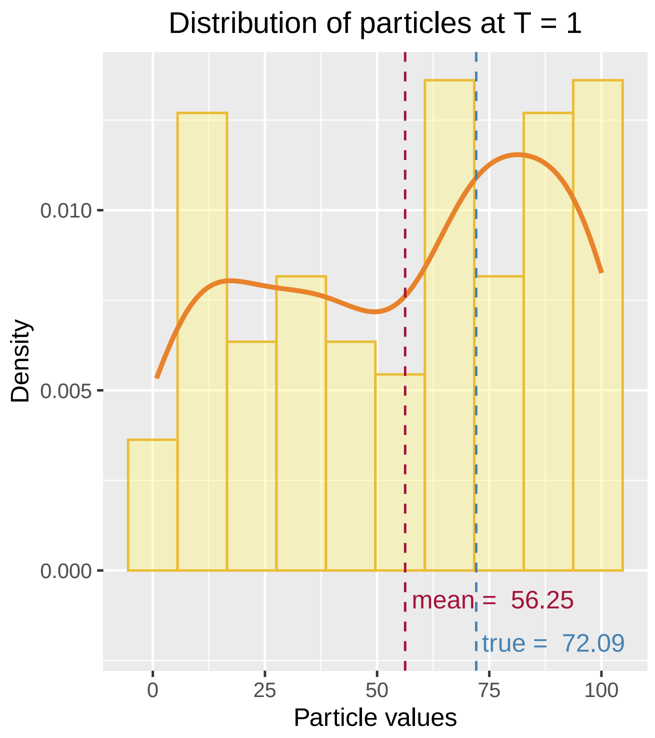


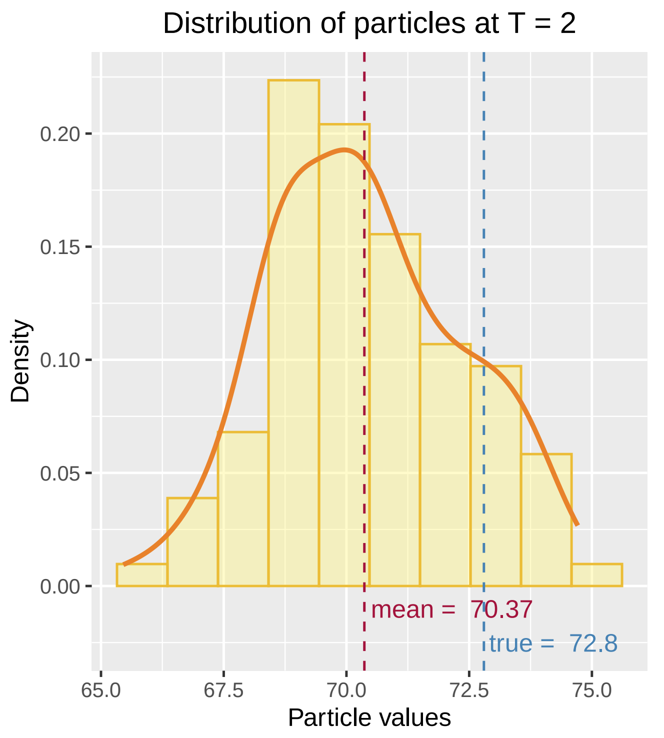


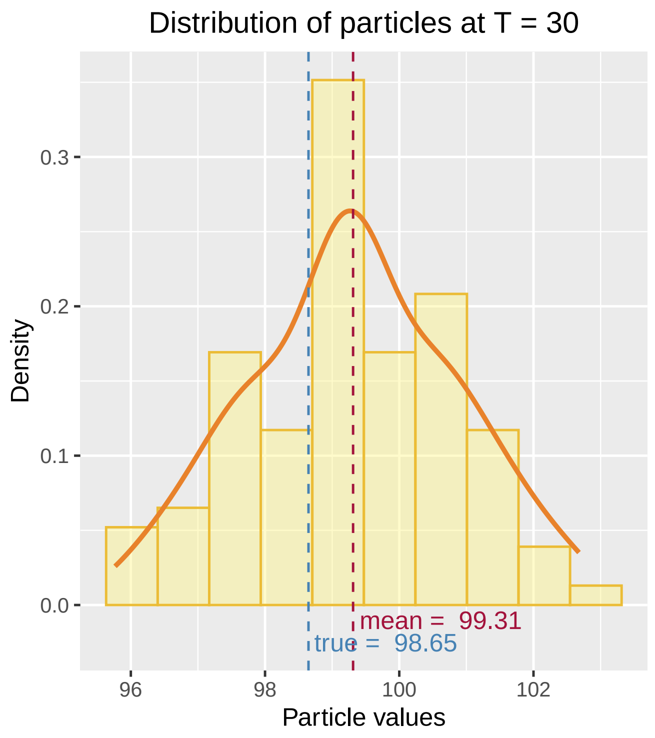
From the above plots, we see that the mean estimates of the particle filter are quite accurate when compared to the true states. From the residuals plot, we can see that the residuals are somewhat normally distributed and that most of the residuals are falling in the range of $[-3, 3]$.

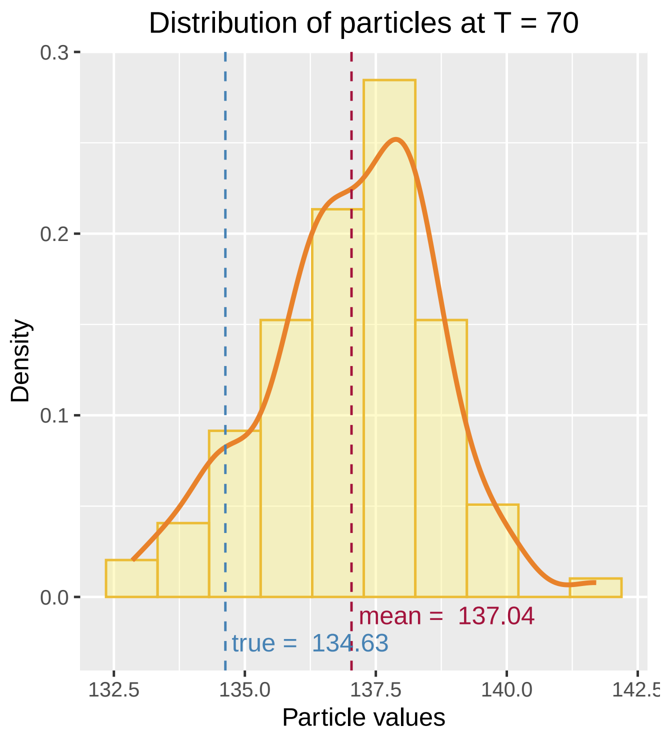
Particle values at different time steps

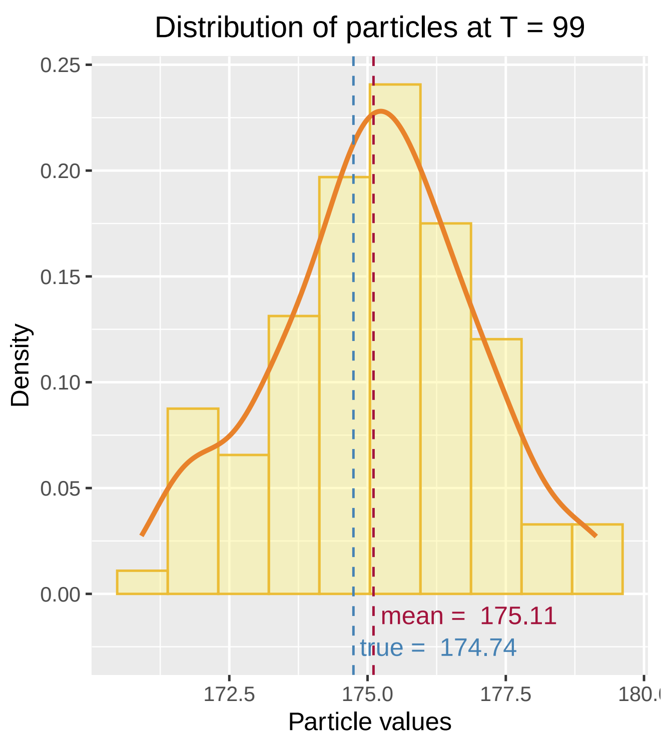
To visualize the particle values at certain time steps, we plot their histogram and density distributions along with their mean and the true state. We show these plots for the first 2 time steps, last 2 time steps and the intermediate time steps - 30 and 70.

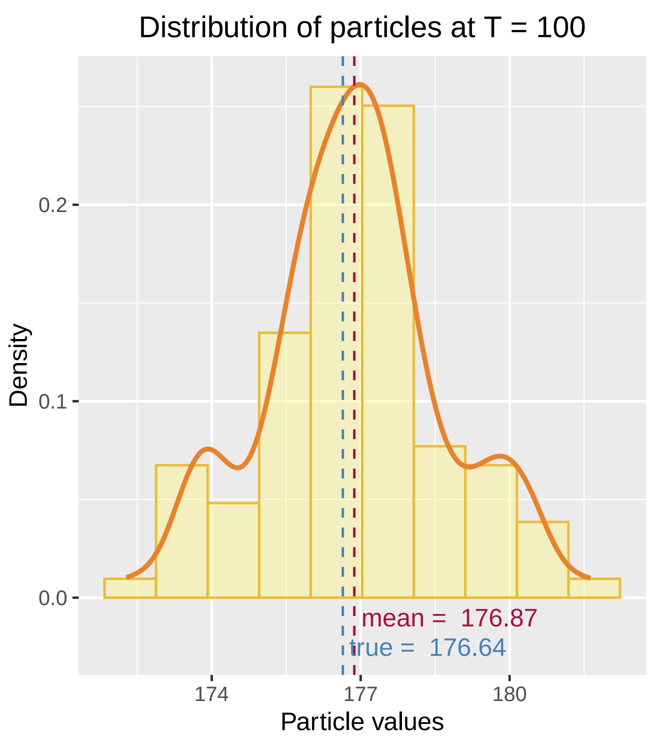












At $T = 0$, we sample the particles from the initial model which is $Unif(0, 100)$ with a mean estimate close to 50. But, after resampling at $T = 1$, we see that the values closer to the observation get higher weights and hence, are sampled more often. So, the mean estimate has moved from 50 towards the observation value. At higher time steps, we see that the distribution is more concentrated in a smaller range of values and the mean estimate is very close to the true state.

Question 2: Particle filters with varied emission model SD

Question: 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:

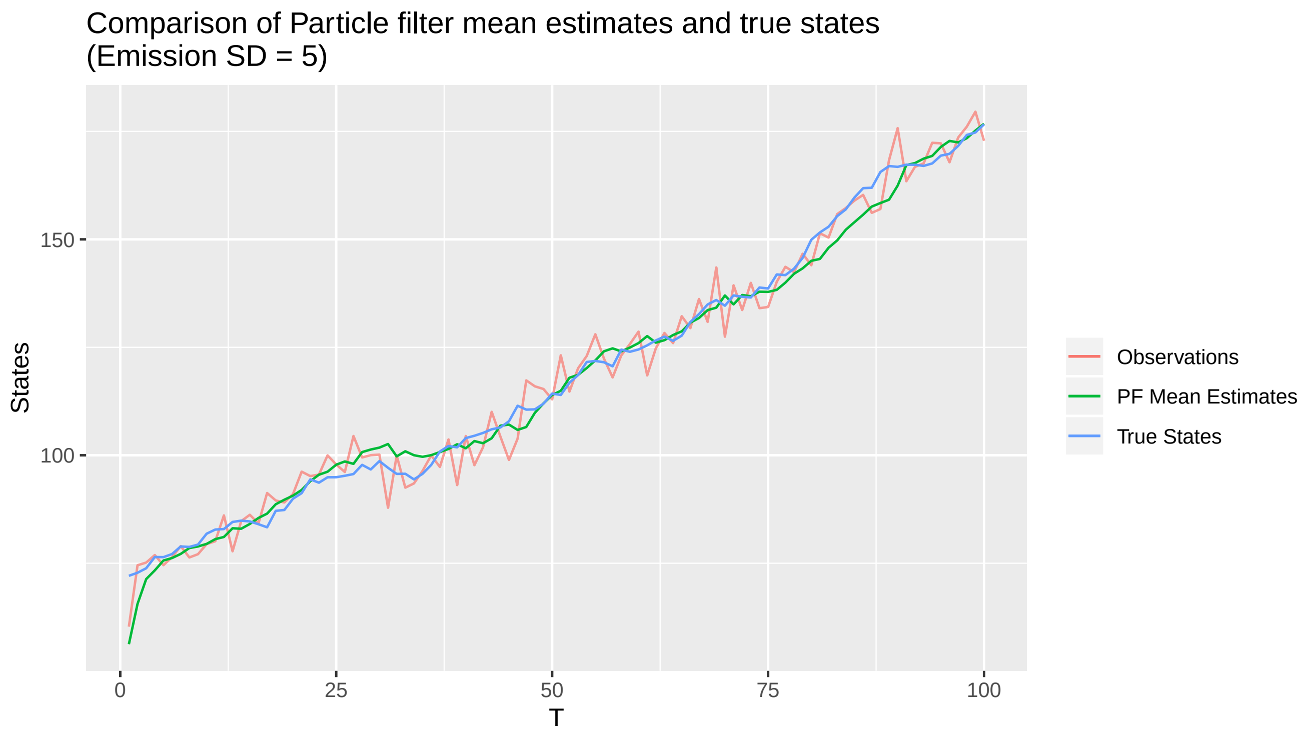
In this task, we want to try the particle filter by varying the emission model with SD $= 5$ and SD $= 50$ (earlier we used SD $= 1$). So, we create these new models and simulate states and observations from the SSMs. Then, we run the particle filter on these observations.

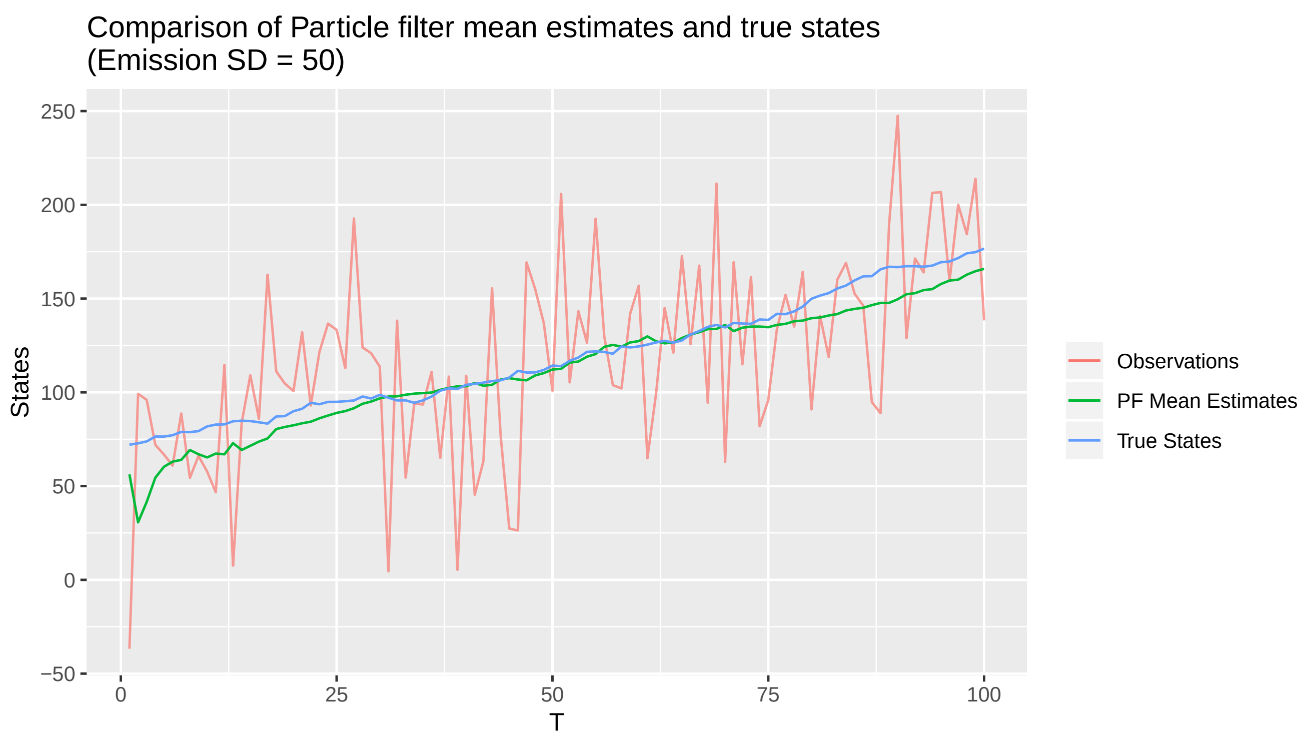
```r
# Emission model with SD = 5
emission_model_sd5 = function(z_t) emission_model(z_t, sd = 5)
emission_prob_sd5 = function(x_t, z_t) emission_prob(x_t, z_t, sd = 5)
```

```r
# Emission model with SD = 50
emission_model_sd50 = function(z_t) emission_model(z_t, sd = 50)
emission_prob_sd50 = function(x_t, z_t) emission_prob(x_t, z_t, sd = 50)
```

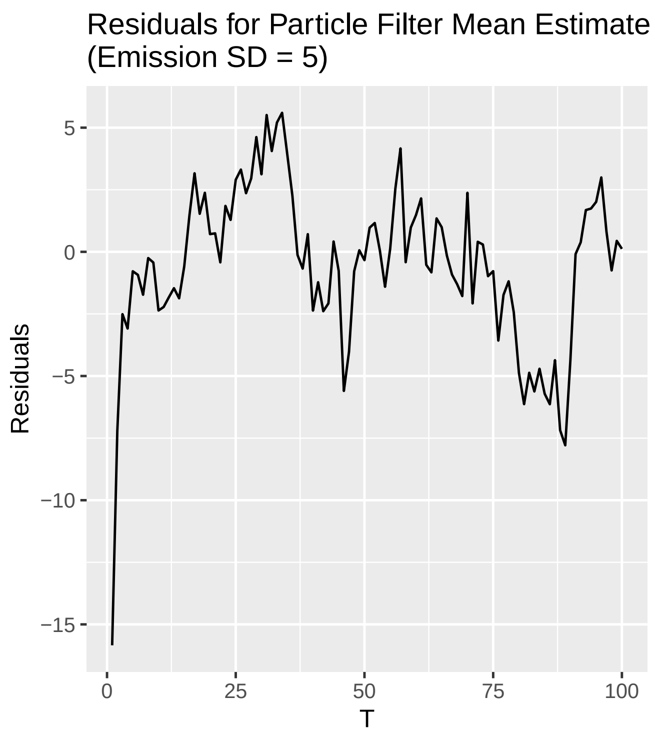
```r
# Running SSM simulation and particle filter with emission model SD = 5
set.seed(12345)
ssm_sd5_sim = simSSM(100, initial_model, transition_model, emission_model_sd5)
ssm_sd5_obs = ssm_sd5_sim$observations
ssm_sd5_states = ssm_sd5_sim$states

pf_sd5_res = particle_filter(100, ssm_sd5_obs, initial_model, transition_model,
                             emission_model_sd5, emission_prob_sd5, method = 2)
residuals_sd5 = pf_sd5_res$pf_means - ssm_sd5_states

# Running SSM simulation and particle filter with emission model SD = 50
set.seed(12345)
ssm_sd50_sim = simSSM(100, initial_model, transition_model, emission_model_sd50)
ssm_sd50_obs = ssm_sd50_sim$observations
ssm_sd50_states = ssm_sd50_sim$states

pf_sd50_res = particle_filter(100, ssm_sd50_obs, initial_model, transition_model,
                              emission_model_sd50, emission_prob_sd50, method = 2)
residuals_sd50 = pf_sd50_res$pf_means - ssm_sd50_states
```

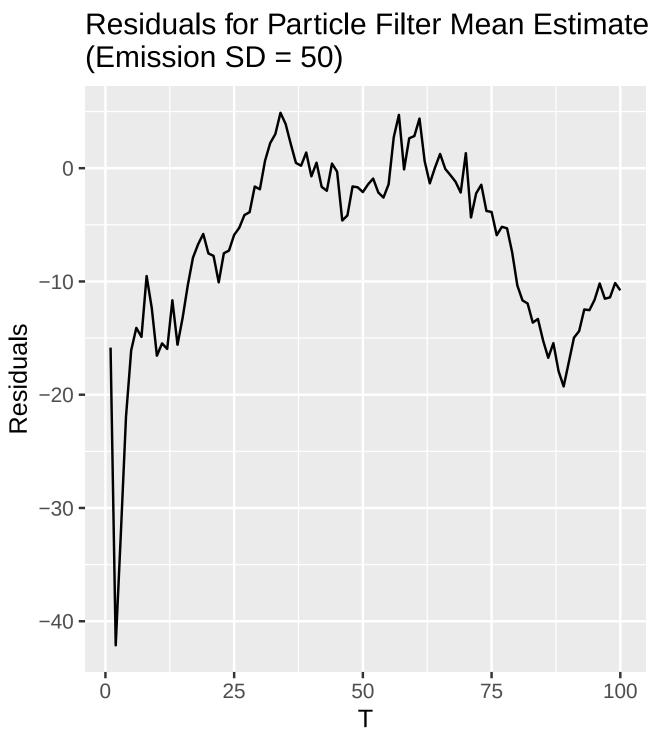
The following plots show the mean estimates of the particle filter, true states and the observations when we used SD = 5 and SD = 50.

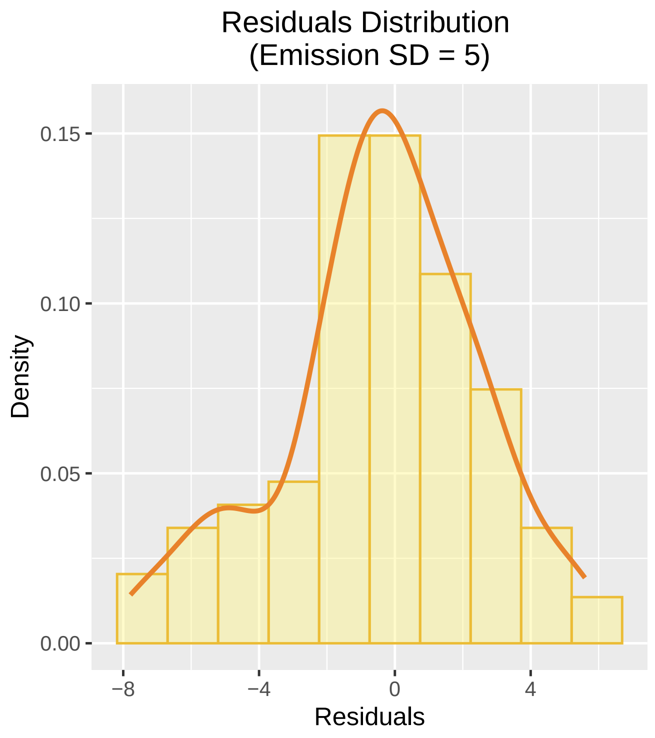


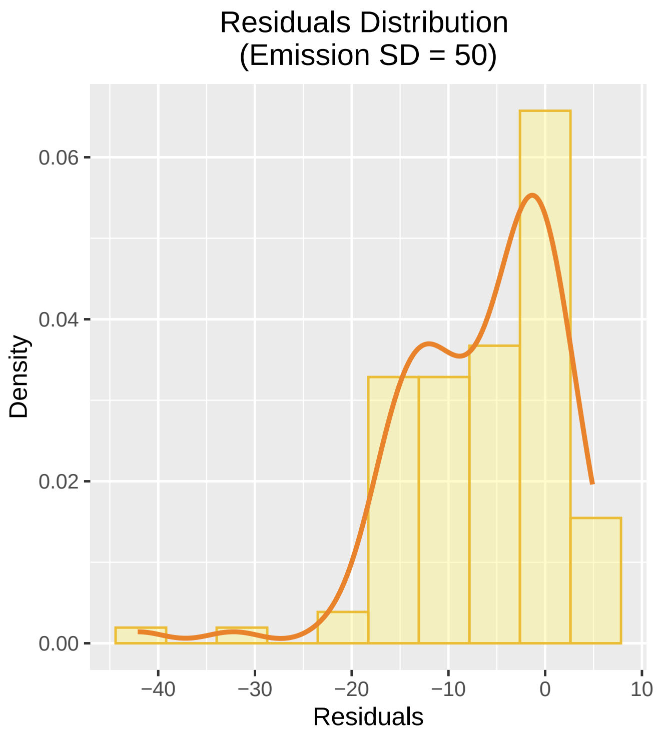


From the above plots, we see as expected that the variation in the observations is higher when we increase the SD of the emission model. The mean estimates of the particles are still very reasonable for SD = 5 but not as accurate as when we used SD = 1. But with SD = 50, the mean estimates do not look reasonable for many of the time steps.









When SD $= 5$, the residuals are mostly in the range $[-8, 5]$ and when SD $= 50$, the residuals are mostly in the range $[-20, 10]$. Comparatively, the residuals were mostly in the range $[-3, 3]$ when SD $= 1$. So, when we increase the SD of the emission model, the particle filter estimates become less accurate. We see that the observations themselves are very noisy and vary a lot compared to the true state when the emission model has higher SD. Since the noise in the emission model is high, it is difficult for the particle filter to estimate the hidden state and hence, it is not very accurate.

Question 3: Particle filter without weight correction

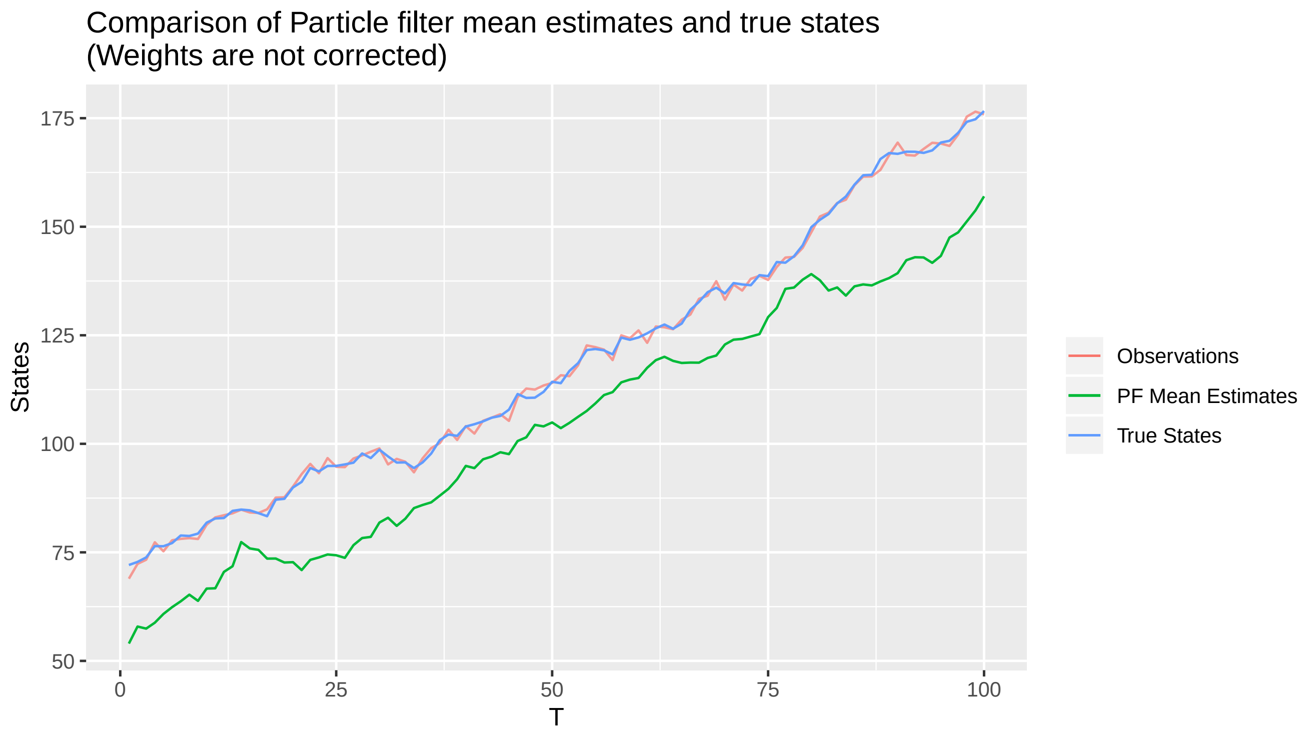
Question: Finally, show and explain what happens when the weights in the particle filter are always equal to 1, i.e. there is no correction.

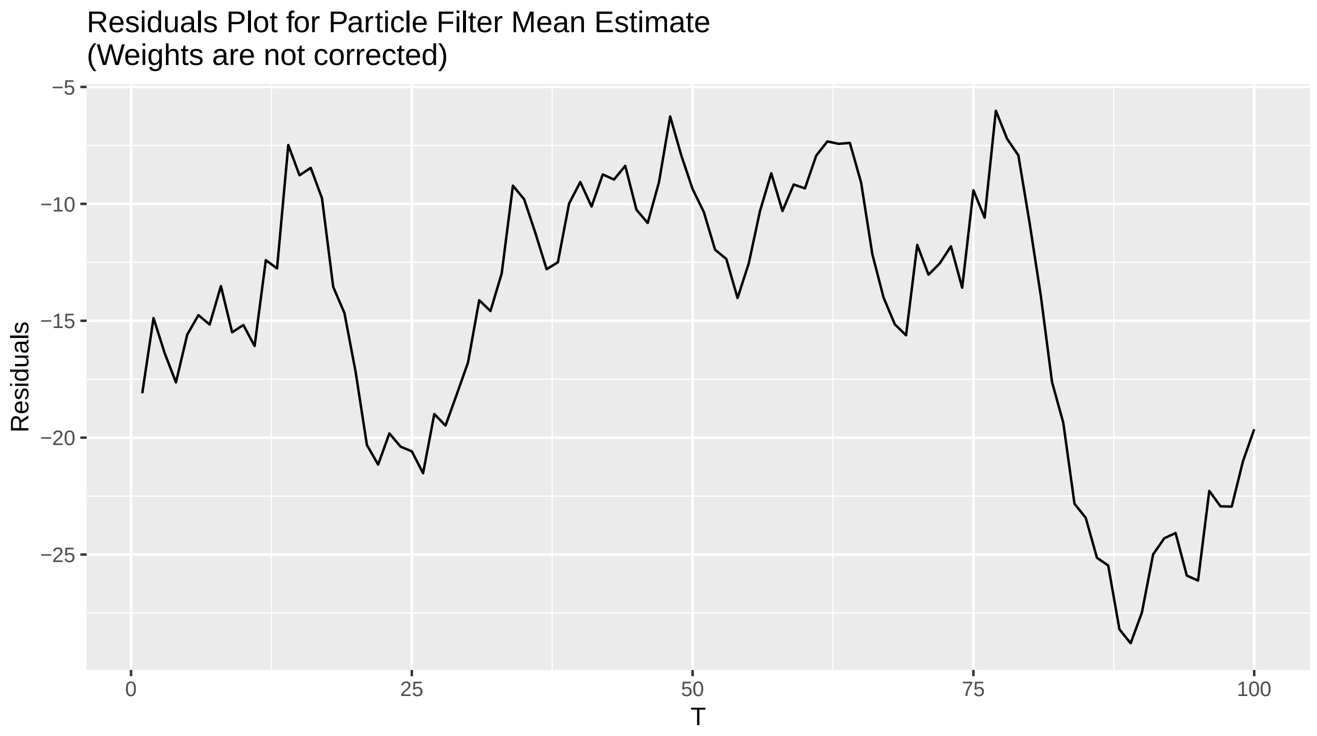
Answer:

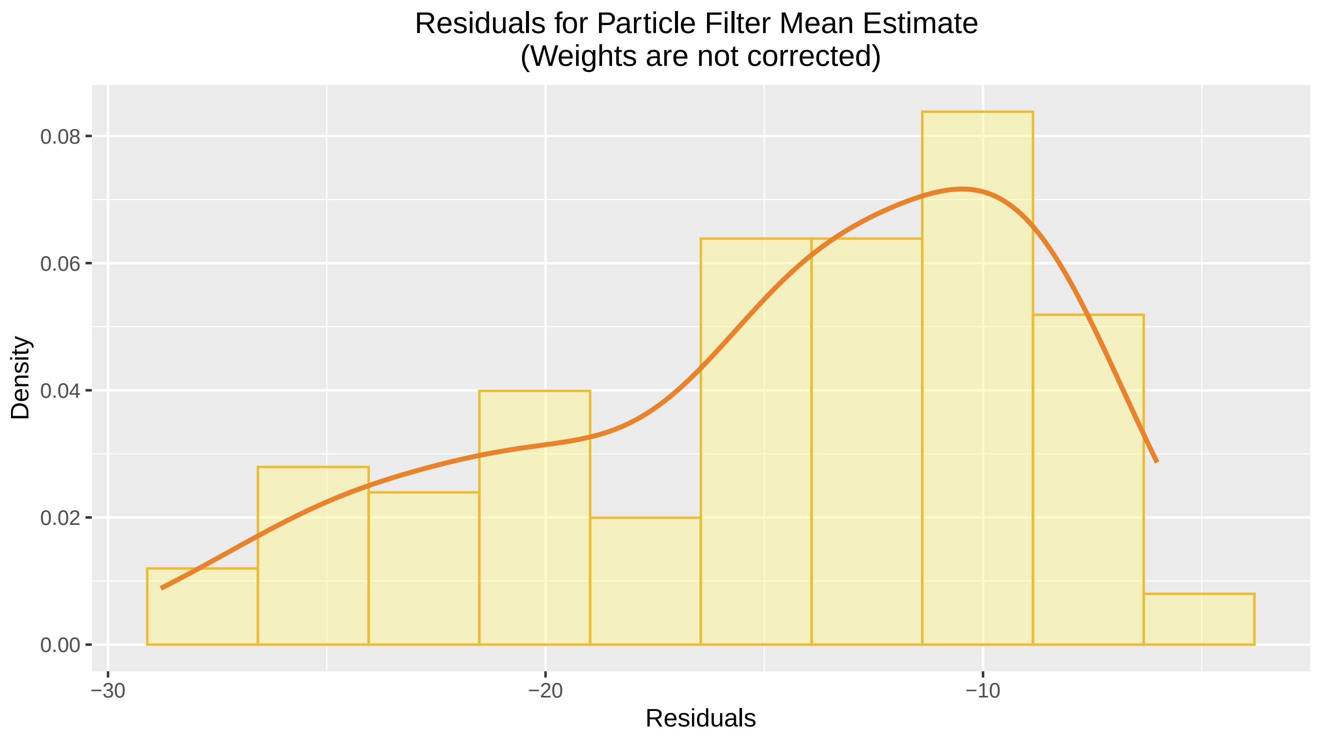
In this task, we run the particle filter using the same simulation of states and observations as in the 1st question but without updating/correcting the weights in the particle filter. So, the weights are all equal to 1 initially and after normalizing, they are equal to 1/(number of particles).

```
# Running particle filter without weight correction
set.seed(123)
pf_uncorr_res = particle_filter(100, ssm_obs, initial_model, transition_model,
                                emission_model, emission_prob, method = 2,
                                weight_update = FALSE)
residuals_uncorr = pf_uncorr_res$pf_means - ssm_states
```

The following plot shows the mean estimates of the particles, the true states (z_t) and the observations(x_t) at each time step.







From the comparison plot, it seems like there is a lag between the mean estimates of the particle filter and the true states/observations. We see that these residuals are in the range of $[-30, -15]$.

Since the weights are all equal, all the particles have an equal probability of being propagated to the next time step. This means that the observations have no impact on the particles and the particle values change only based on the transition model. We initially sample the particles from the initial model which is $Unif(0, 100)$ and it has a mean estimate close to 50. The first true state has a value around 70 and hence, the mean estimate varies by around -20. Since, we do not correct/update the weights, even those particles which would be less likely given the observation at that time step get propagated further and leads to this error being propagated to successive time steps.

Appendix

```r
# Set up general options

knitr::opts_chunk$set(echo = FALSE, warning = FALSE, message = FALSE, fig.width=8)

set.seed(123456)

library(ggplot2)
options(kableExtra.latex.load_packages = FALSE)
library(kableExtra)
library(caret)
library(gridExtra)

options(scipen=999)

# Function to plot histogram and density of a sample
hist_plt = function(vals){
  n_bins = ceiling(sqrt(length(vals)))
  plt = qplot(vals, geom = 'blank') +
    geom_histogram(aes(y = ..density..), alpha = 0.4,
                   bins = n_bins, color = "#EBBC36", fill = "#FFF57D") +
    geom_line(aes(y = ..density.., color = 'Empirical Density'),
              stat = 'density', size = 1, color = "#E8822B") +
    theme(plot.title = element_text(hjust = 0.5),
          plot.subtitle = element_text(hjust = 0.5)) + ylab("Density")

  return(plt)
}


# --------------------------------------------------------------------------------
# Question 1
# --------------------------------------------------------------------------------

# Function to simulate states and observations for a State Space Model
# using the given initial model, transition model and emission model
simSSM = function(N, initial_model, transition_model, emission_model){
  z_t = vector(mode = "numeric", length = N)

  # Simulate initial state
  z_t[1] = initial_model()

  # Simulate remaining states
  for(i in 2:N){
    z_t[i] = transition_model(z_t[i-1])
  }

  # Simulate observations from states
  x_t = sapply(z_t, emission_model)

  res = list(states = z_t, observations = x_t)

  return(res)
}

# Transition model - mixture of 3 gaussians
transition_model = function(z_t_1){
  mean_z_t = sample(c(z_t_1, z_t_1+1, z_t_1+2), size = 1, prob = rep(1/3, 3))
  return(rnorm(1, mean_z_t, 1))
}

# Emission model - mixture of 3 gaussians (SD = 1)
emission_model = function(z_t, sd = 1){
  mean_x_t = sample(c(z_t, z_t+1, z_t-1), size = 1, prob = rep(1/3, 3))
  return(rnorm(1, mean_x_t, sd))
}

# Initial model - Unif(0, 100)
initial_model = function(){
  return(runif(1, min = 0, max = 100))
}

# Probability function for emission model
emission_prob = function(x_t, z_t, sd = 1){
  return((dnorm(x_t, mean = z_t, sd = sd) +
            dnorm(x_t, mean = z_t-1, sd = sd) +
            dnorm(x_t, mean = z_t+1, sd = sd)) / 3)
}

# Simulate 100 time steps of SSM
set.seed(12345)
ssm_sim = simSSM(100, initial_model, transition_model, emission_model)
ssm_obs = ssm_sim$observations
```

```r
ssm_states = ssm_sim$states

# Function to perform particle filtering
particle_filter = function(num_particles, observations, initial_model,
                           transition_model, emission_model, emission_prob,
                           method = 2, weight_update = TRUE){
  # Number of time steps
  N = length(observations)

  # Matrix to store the particles at all time steps
  particle_history = matrix(nrow = N, ncol = num_particles)

  # Generate initial particles from initial model
  particles = sapply(1:num_particles, function(i) initial_model())

  # Weights
  w_t = rep(1, num_particles)

  for(t in 1:N){
    # Method 1
    if(method == 1){
      # Vector for belief set of particles
      bel_particles = vector(mode = "numeric", length = num_particles)

      for(j in 1:num_particles){
        # Generate belief particle using transition model
        # z_t^m given z_t-1
        bel_particles[j] = transition_model(particles[j])

        if(weight_update){
          # Calculate weight using emission model probability
          # probability of x_t given z_t^m
          w_t[j] = emission_prob(observations[t], bel_particles[j])
        }
      }

      # Normalize weights
      w_t = w_t/sum(w_t)

      # Resample particles from belief set of particles
      # using weighted probabilities with replacement
      particles = sample(bel_particles, size = num_particles,
                         prob = w_t, replace = TRUE)
    }
    # Method 2
    else{
      # Sample particles from the mixture model of weighted transition models

      # Sample component mean from particles at time t-1
      # using weighted probabilities with replacement
      # z_(t-1)^m given x_t-1
      post_sample = sample(particles, size = num_particles,
                           prob = w_t, replace = TRUE)

      for(j in 1:num_particles){
        # Resample particles based on transition model and sampled component mean
        # z_t given z_t-1
        particles[j] = transition_model(post_sample[j])

        if(weight_update){
          # Calculate weight using emission model probability
          # probability of x_t given z_t
          w_t[j] = emission_prob(observations[t], particles[j])
        }
      }

      # Normalize weights
      w_t = w_t/sum(w_t)
    }

    # Add the set of particles to particle history
    particle_history[t, ] = particles
  }

  # Calculate mean estimates of particles for each time step
  mean_estimates = rowMeans(particle_history)

  res = list(pf_history = particle_history,
             pf_means = mean_estimates)

  return(res)
}

# Running particle filter
pf_res = particle_filter(100, ssm_obs, initial_model, transition_model,
                         emission_model, emission_prob, method = 2)
residuals = pf_res$pf_means - ssm_states

ggplot() +
  geom_line(aes(x = 1:100, y = ssm_obs, color = "Observations"), alpha = 0.7) +
  geom_line(aes(x = 1:100, y = pf_res$pf_means, color = "PF Mean Estimates")) +
  geom_line(aes(x = 1:100, y = ssm_states, color = "True States")) +
```

```r
  labs(x = "T", y = "States", color = "") +
  ggtitle("Comparison of Particle filter mean estimates and true states")

ggplot() + geom_line(aes(x = 1:100, y = residuals)) +
  ggtitle("Residuals Plot for Particle Filter Mean Estimate") +
  labs(x = "Time (T)", y = "Residuals")

hist_plt(residuals[2:length(residuals)]) +
  ggtitle("Residuals for Particle Filter Mean Estimate") +
  labs(x = "Residuals")
# Function to plot distribution of particles at a given time step
# Also, plots lines to show particles mean and true state
plot_particle_dist = function(t, pf_history, pf_means, true_states,
                              annotation_y = c(-0.01, -0.025)){
  hist_plt(pf_history[t,]) +
    geom_vline(xintercept = pf_means[t], color = "#A3133C", linetype = "dashed") +
    annotate("text", x = pf_means[t], y = annotation_y[1],
             label = paste("mean = ", round(pf_means[t], 2)),
             color = "#A3133C", hjust = -0.04) +
    geom_vline(xintercept = true_states[t], color = "steelblue",
               linetype = "dashed") +
    annotate("text", x = true_states[t], y = annotation_y[2],
             label = paste("true = ", round(true_states[t], 2)),
             color = "steelblue", hjust = -0.04) +
    ggtitle(paste("Distribution of particles at T =", t)) +
    labs(x = "Particle values")
}

p1 = plot_particle_dist(1, pf_res$pf_history, pf_res$pf_means,
                        ssm_states, annotation_y = c(-0.0008, -0.002))
p2 = plot_particle_dist(2, pf_res$pf_history, pf_res$pf_means, ssm_states)
grid.arrange(p1, p2, nrow = 1)

p3 = plot_particle_dist(30, pf_res$pf_history, pf_res$pf_means, ssm_states)
p4 = plot_particle_dist(70, pf_res$pf_history, pf_res$pf_means, ssm_states)
grid.arrange(p3, p4, nrow = 1)

p5 = plot_particle_dist(99, pf_res$pf_history, pf_res$pf_means, ssm_states)
p6 = plot_particle_dist(100, pf_res$pf_history, pf_res$pf_means, ssm_states)
grid.arrange(p5, p6, nrow = 1)

# ---------------------------------------------------------------------------
# Question 2
# ---------------------------------------------------------------------------

# Emission model with SD = 5
```

```r
emission_model_sd5 = function(z_t) emission_model(z_t, sd = 5)
emission_prob_sd5 = function(x_t, z_t) emission_prob(x_t, z_t, sd = 5)

# Emission model with SD = 50
emission_model_sd50 = function(z_t) emission_model(z_t, sd = 50)
emission_prob_sd50 = function(x_t, z_t) emission_prob(x_t, z_t, sd = 50)

# Running SSM simulation and particle filter with emission model SD = 5
set.seed(12345)
ssm_sd5_sim = simSSM(100, initial_model, transition_model, emission_model_sd5)
ssm_sd5_obs = ssm_sd5_sim$observations
ssm_sd5_states = ssm_sd5_sim$states

pf_sd5_res = particle_filter(100, ssm_sd5_obs, initial_model, transition_model,
                             emission_model_sd5, emission_prob_sd5, method = 2)
residuals_sd5 = pf_sd5_res$pf_means - ssm_sd5_states

# Running SSM simulation and particle filter with emission model SD = 50
set.seed(12345)
ssm_sd50_sim = simSSM(100, initial_model, transition_model, emission_model_sd50)
ssm_sd50_obs = ssm_sd50_sim$observations
ssm_sd50_states = ssm_sd50_sim$states

pf_sd50_res = particle_filter(100, ssm_sd50_obs, initial_model, transition_model,
                              emission_model_sd50, emission_prob_sd50, method = 2)
residuals_sd50 = pf_sd50_res$pf_means - ssm_sd50_states

ggplot() +
  geom_line(aes(x = 1:100, y = ssm_sd5_obs, color = "Observations"), alpha = 0.7) +
  geom_line(aes(x = 1:100, y = pf_sd5_res$pf_means, color = "PF Mean Estimates")) +
  geom_line(aes(x = 1:100, y = ssm_sd5_states, color = "True States")) +
  labs(x = "T", y = "States", color = "") +
  ggtitle(paste("Comparison of Particle filter mean estimates and true states",
                "\n(Emission SD = 5)"))

ggplot() +
  geom_line(aes(x = 1:100, y = ssm_sd50_obs, color = "Observations"), alpha = 0.7) +
  geom_line(aes(x = 1:100, y = pf_sd50_res$pf_means, color = "PF Mean Estimates")) +
  geom_line(aes(x = 1:100, y = ssm_sd50_states, color = "True States")) +
  labs(x = "T", y = "States", color = "") +
  ggtitle(paste("Comparison of Particle filter mean estimates and true states",
                "\n(Emission SD = 50)"))

res_plt_sd5 = ggplot() + geom_line(aes(x = 1:100, y = residuals_sd5)) +
  ggtitle(paste("Residuals for Particle Filter Mean Estimate",
                "\n(Emission SD = 5)")) +
  labs(x = "T", y = "Residuals")

res_plt_sd50 = ggplot() + geom_line(aes(x = 1:100, y = residuals_sd50)) +
  ggtitle(paste("Residuals for Particle Filter Mean Estimate",
                "\n(Emission SD = 50)")) +
  labs(x = "T", y = "Residuals")

grid.arrange(res_plt_sd5, res_plt_sd50, nrow = 1)

res_dist_sd5 = hist_plt(residuals_sd5[2:100]) +
  ggtitle(paste("Residuals Distribution", "\n(Emission SD = 5)")) +
  labs(x = "Residuals")

res_dist_sd50 = hist_plt(residuals_sd50[2:100]) +
  ggtitle(paste("Residuals Distribution", "\n(Emission SD = 50)")) +
  labs(x = "Residuals")

grid.arrange(res_dist_sd5, res_dist_sd50, nrow = 1)

# ------------------------------------------------------------------------------
# Question 3
# ------------------------------------------------------------------------------

# Running particle filter without weight correction
set.seed(123)
pf_uncorr_res = particle_filter(100, ssm_obs, initial_model, transition_model,
                                emission_model, emission_prob, method = 2,
                                weight_update = FALSE)
residuals_uncorr = pf_uncorr_res$pf_means - ssm_states

ggplot() +
  geom_line(aes(x = 1:100, y = ssm_obs, color = "Observations"), alpha = 0.7) +
  geom_line(aes(x = 1:100, y = pf_uncorr_res$pf_means, color = "PF Mean Estimates")) +
  geom_line(aes(x = 1:100, y = ssm_states, color = "True States")) +
  labs(x = "T", y = "States", color = "") +
  ggtitle(paste("Comparison of Particle filter mean estimates and true states",
                "\n(Weights are not corrected)"))

ggplot() + geom_line(aes(x = 1:100, y = residuals_uncorr)) +
  ggtitle(paste("Residuals Plot for Particle Filter Mean Estimate",
                "\n(Weights are not corrected)")) +
  labs(x = "T", y = "Residuals")

hist_plt(residuals_uncorr[2:100]) + labs(x = "Residuals") +
  ggtitle(paste("Residuals for Particle Filter Mean Estimate",
                "\n(Weights are not corrected)"))
```