Advanced Machine Learning

Lab 4

Rasmus Holm

2017-10-08

Assignment 1

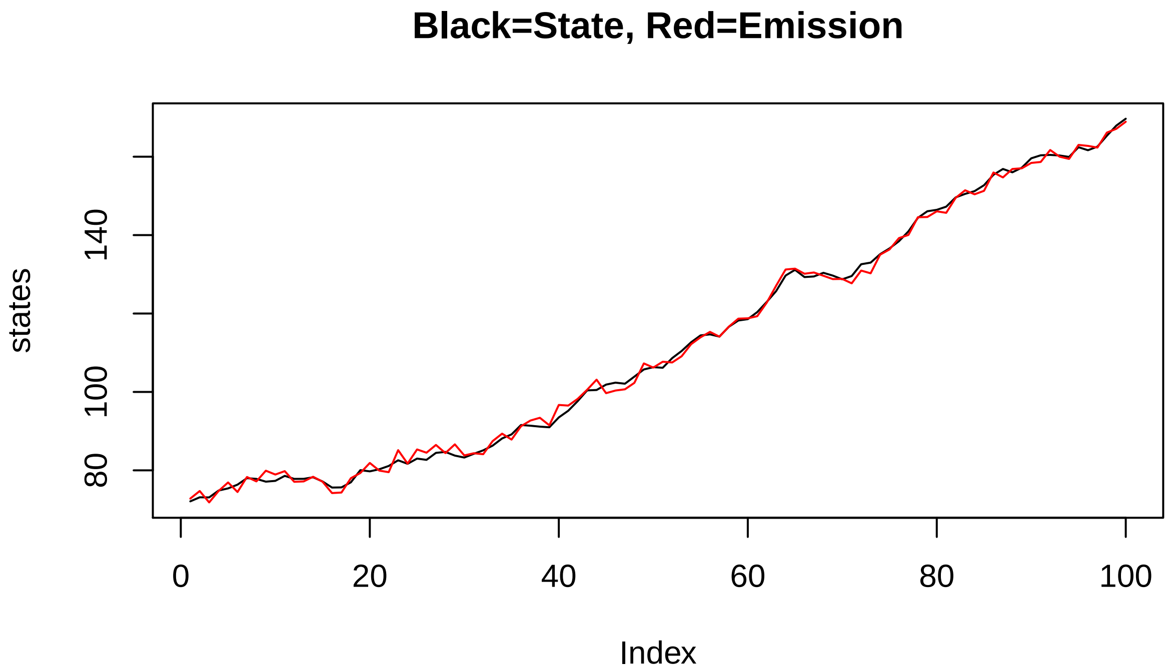
```
sample_transition_component <- function() {
    sample(1:3, size=1, prob=rep(1, 3) / 3)
}

sample_emission_component <- function() {
    sample(1:3, size=1, prob=rep(1, 3) / 3)
}

sample_initial_state <- function() {
    runif(1, 0, 100)
}

sample_transition <- function(z, sigma=1) {
    component <- sample_transition_component()
    mu <- ifelse(component == 1, z, ifelse(component == 2, z + 1, z + 2))
    rnorm(1, mean=mu, sd=sigma)
}

sample_emission <- function(z, sigma=1) {
    component <- sample_emission_component()
    mu <- ifelse(component == 1, z, ifelse(component == 2, z - 1, z + 1))
    rnorm(1, mean=mu, sd=sigma)
}

generate_states_and_emissions <- function(n, sigma) {
    states <- rep(0, n)
    emissions <- rep(0, n)

    initial_state <- sample_initial_state()
    current_state <- initial_state

    for (i in 1:n) {
        current_emission <- sample_emission(current_state, sigma)

        states[i] <- current_state
        emissions[i] <- current_emission

        current_state <- sample_transition(current_state, sigma)
    }

    list(states=states, emissions=emissions)
}
```

```r
get_emission_density_given_state <- function(emission, state, sigma=1) {
    (dnorm(emission, mean=state, sd=sigma) +
     dnorm(emission, mean=state - 1, sd=sigma) +
     dnorm(emission, mean=state + 1, sd=sigma)) / 3
}

get_sampling_weights <- function(emission, particles, sigma=1) {
    unnormalized_weights <- sapply(particles, function(state) {
        get_emission_density_given_state(emission, state, sigma)
    })
    unnormalized_weights / sum(unnormalized_weights)
}

particle_filtering <- function(x, nparticles, sigma) {
    steps <- length(x)
    particles <- matrix(NA, nrow=steps, ncol=nparticles)
    weights <- matrix(NA, nrow=steps, ncol=nparticles)

    particles[1, ] <- runif(nparticles, 0, 100)
    weights[1, ] <- get_sampling_weights(x[1], particles[1, ], sigma=sigma)

    for (i in 2:steps) {
        predictions <- sample(particles[i - 1, ], size=nparticles,
                              replace=TRUE, prob=weights[i - 1, ])
        particles[i, ] <- sapply(predictions, sample_transition, sigma=sigma)
        weights[i, ] <- get_sampling_weights(x[i], particles[i, ], sigma=sigma)
    }

    list(particles=particles, weights=weights)
}

get_expected_states <- function(particles, weights) {
    rowSums(weights * particles)
}

plot_everything <- function(states, emissions, particles, expected) {
    nparticles <- ncol(particles)

    old <- par(mfrow=c(2, 1))
    plot(states, type="l",
         main="Black=State, Red=Emission",
         ylim=c(min(c(states, emissions)), max(c(states, emissions))))
    lines(emissions, col="red")
    plot(1:nparticles, states, type="l",
         main="Black=State, Orange=Expected",
         ylim=c(min(c(states, expected)), max(c(states, expected))))
    lines(1:nparticles, expected, col="orange")
    par(old)

    plot_helper <- function(idx) {
        p <- particles[idx,]
        ex <- expected[idx]
        s <- states[idx]
```

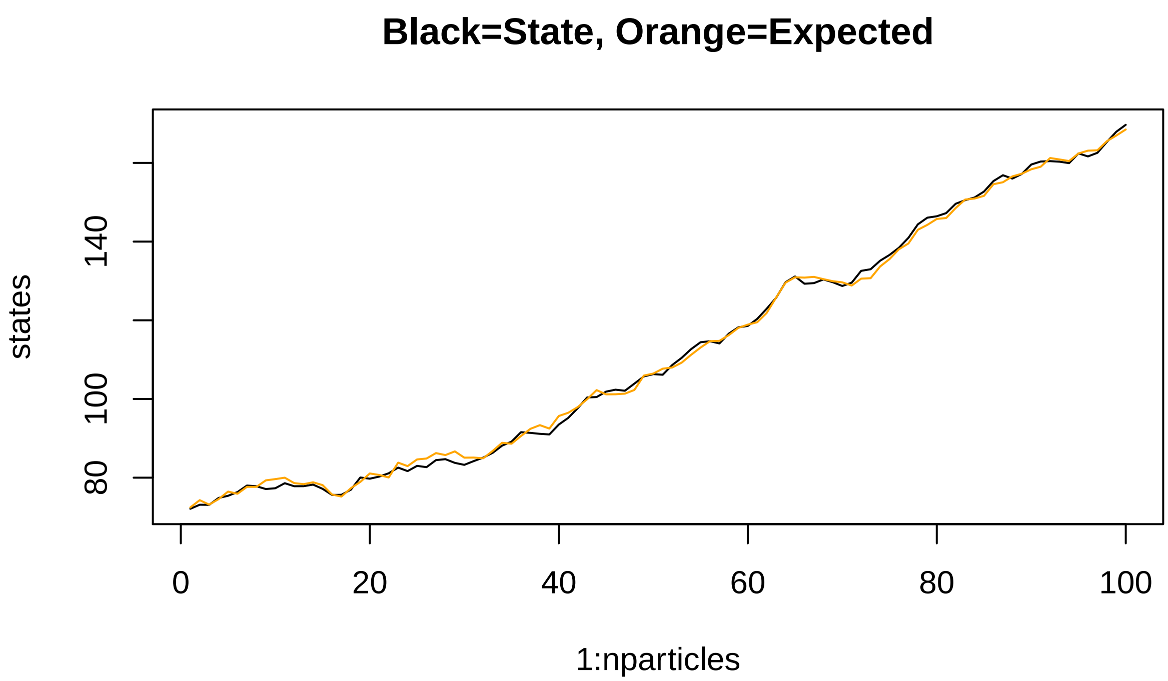
```
        em <- emissions[idx]

        hist(p, breaks=20,
             main=paste("Step", idx, sep=" "),
             xlab="val", xlim=c(min(c(p, ex, s, em)), max(c(p, ex, s, em))))
        abline(v=s, col="blue", lwd=2)
        abline(v=em, col="red", lwd=2)
        abline(v=ex, col="orange", lwd=2)
    }

    old <- par(mfrow=c(2, 2), oma=c(0, 0, 2, 0))
    plot_helper(1)
    plot_helper(33)
    plot_helper(66)
    plot_helper(100)
    title(main="Red=State, Blue=Emission, Orange=Expected",outer=T)
    par(old)
}
```

Sigma 1

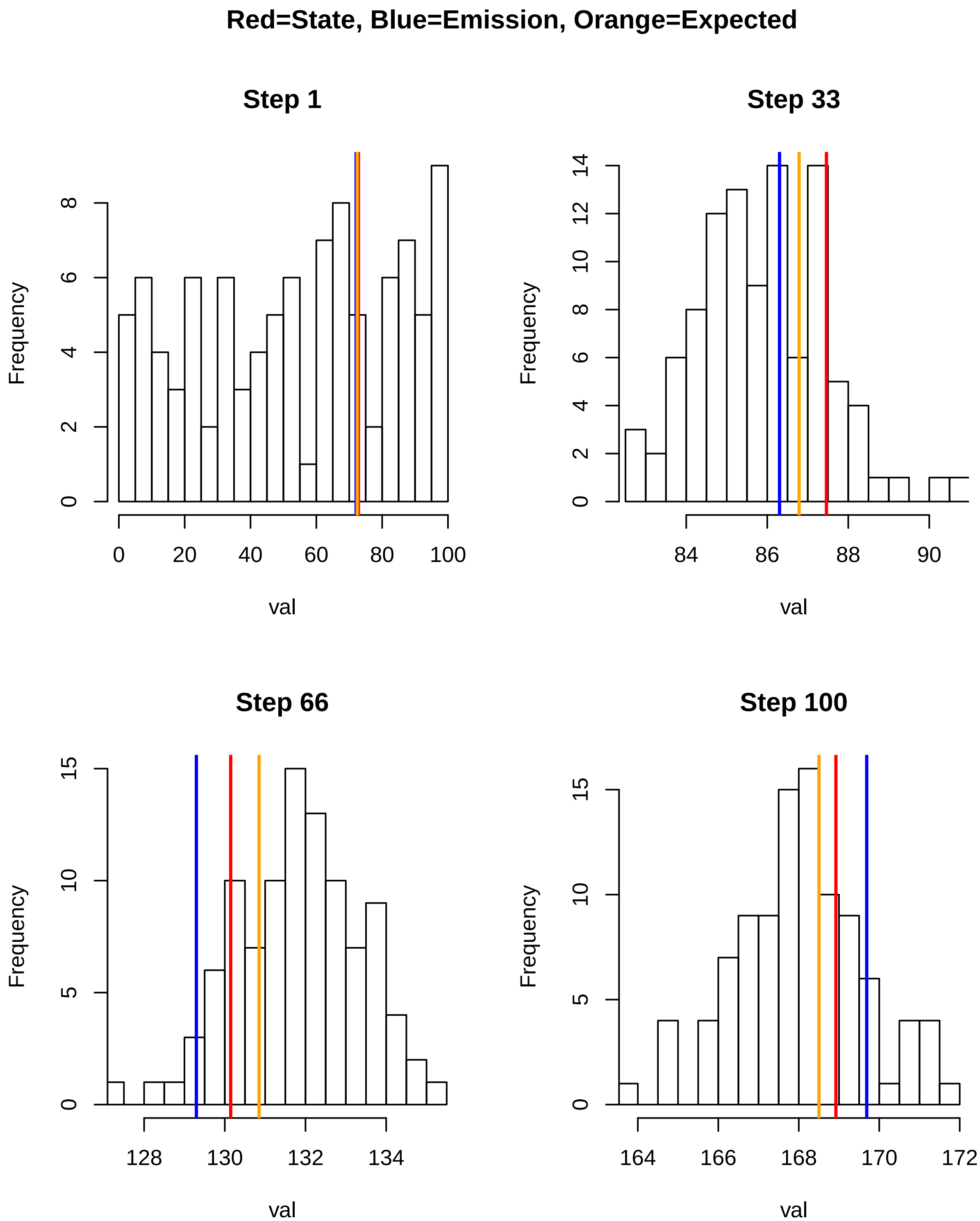
```r
sigma <- 1

set.seed(12345)
samples <- generate_states_and_emissions(100, sigma)

nparticles <- 100
particles <- particle_filtering(samples$emissions, nparticles, sigma)

expected <- get_expected_states(particles$particles, particles$weights)
```



Black=State, Red=Emission
states
80
100
140
0
20
40
60
80
100
Index

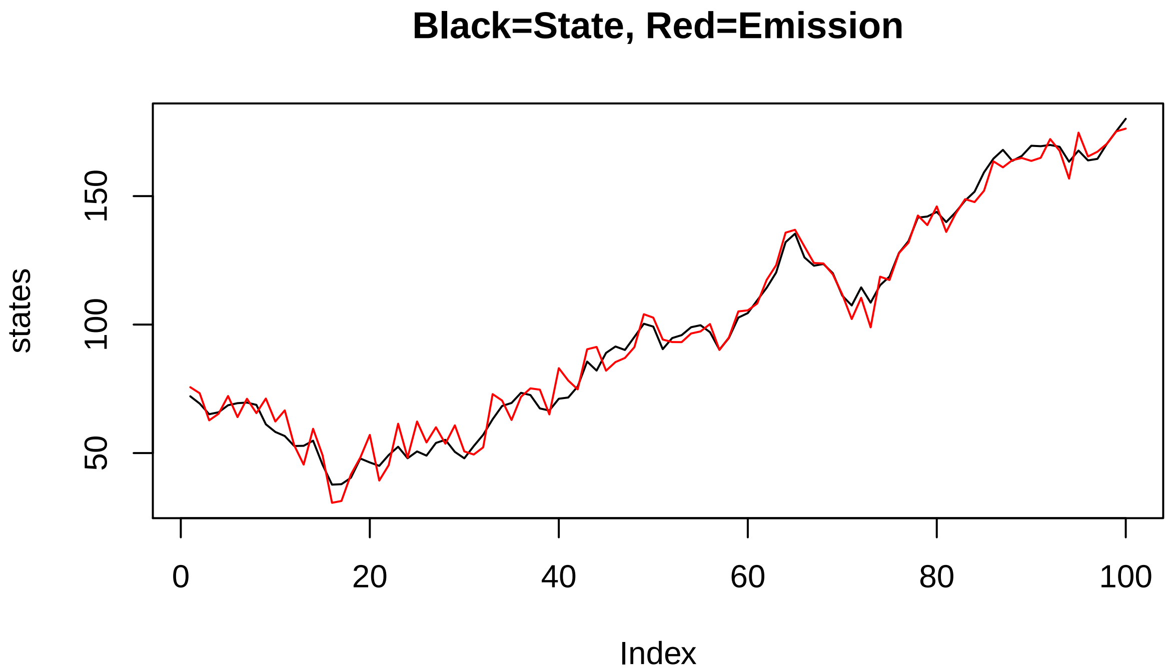


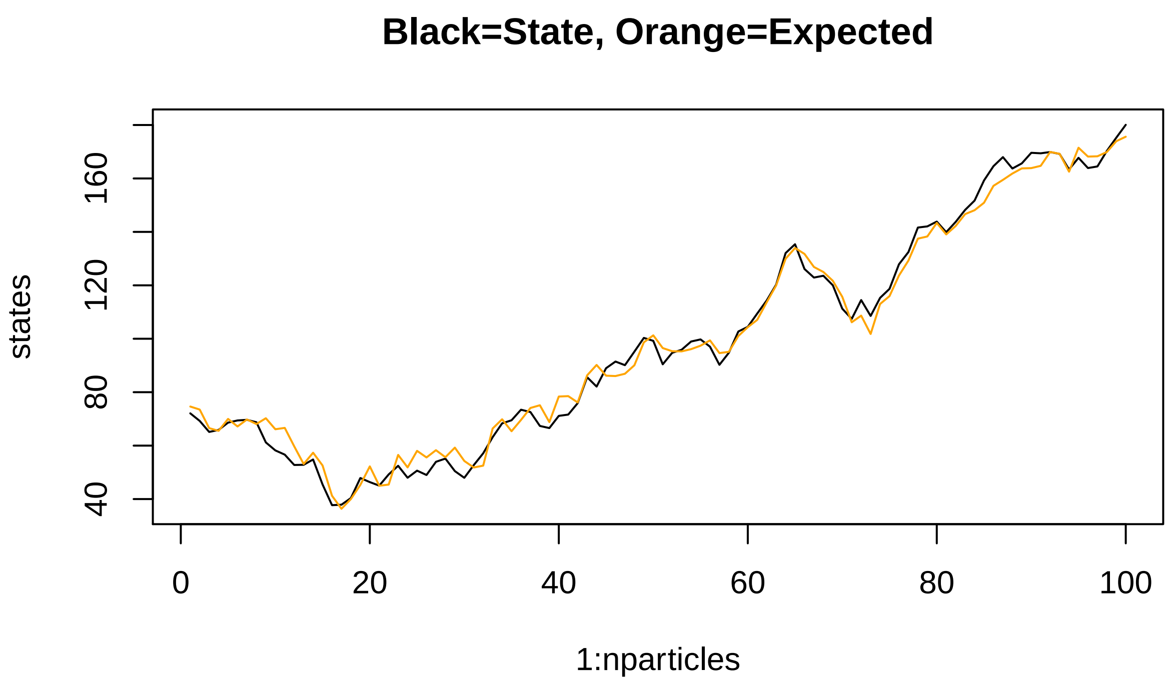
Black=State, Orange=Expected
states
80
100
140
0
20
40
60
80
100
1:nparticles

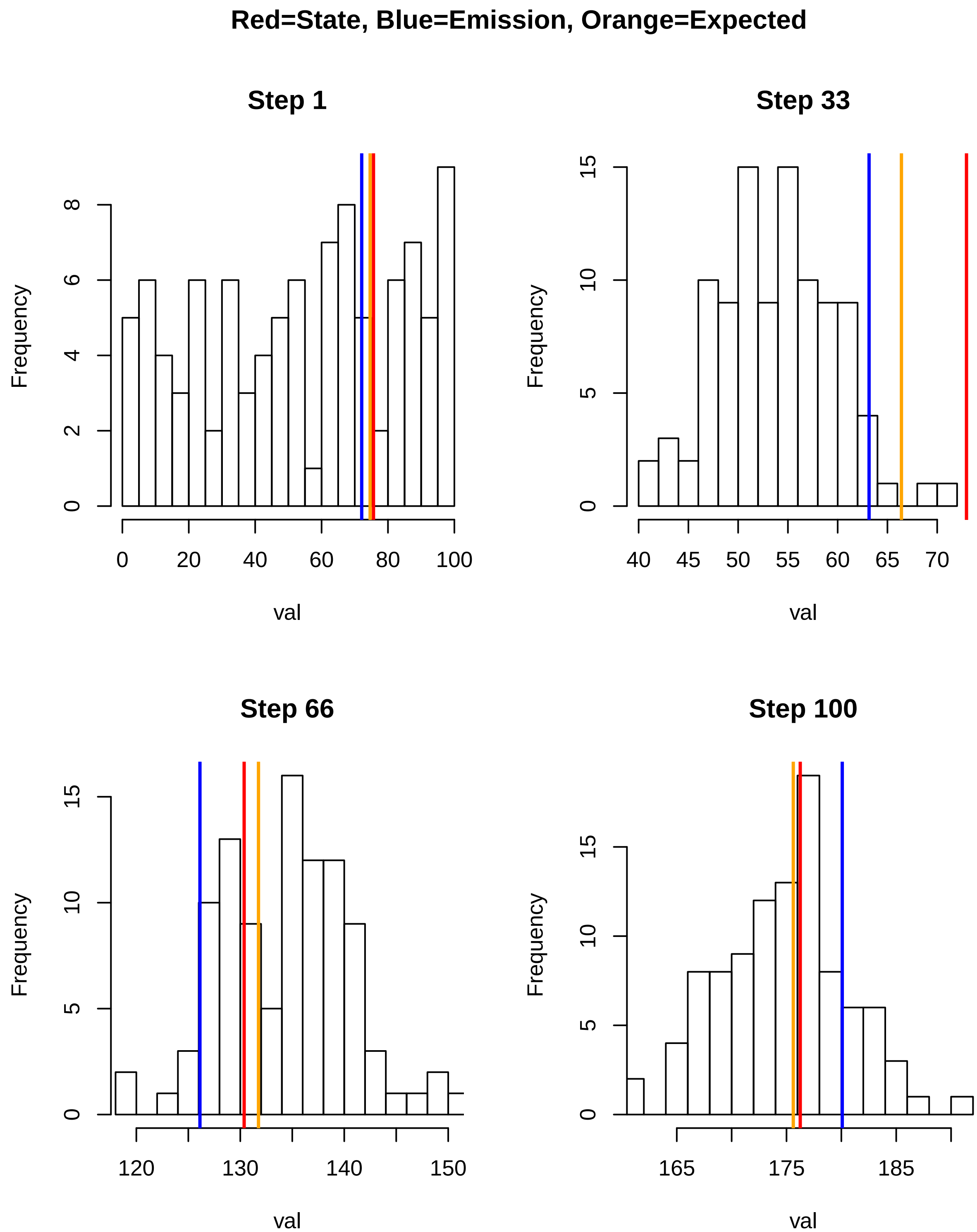

Red=State, Blue=Emission, Orange=Expected
Step 1
Frequency
0
2
4
6
8
0
20
40
60
80
100
val
Step 33
Frequency
0
2
4
6
8
10
12
14
84
86
88
90
val
Step 66
Frequency
0
5
10
15
128
130
132
134
val
Step 100
Frequency
0
5
10
15
164
166
168
170
172
val

Sigma 5

```r
sigma <- 5

set.seed(12345)
samples <- generate_states_and_emissions(100, sigma)

nparticles <- 100
particles <- particle_filtering(samples$emissions, nparticles, sigma)

expected <- get_expected_states(particles$particles, particles$weights)
```


Black=State, Red=Emission
states
50
100
150
0
20
40
60
80
100
Index

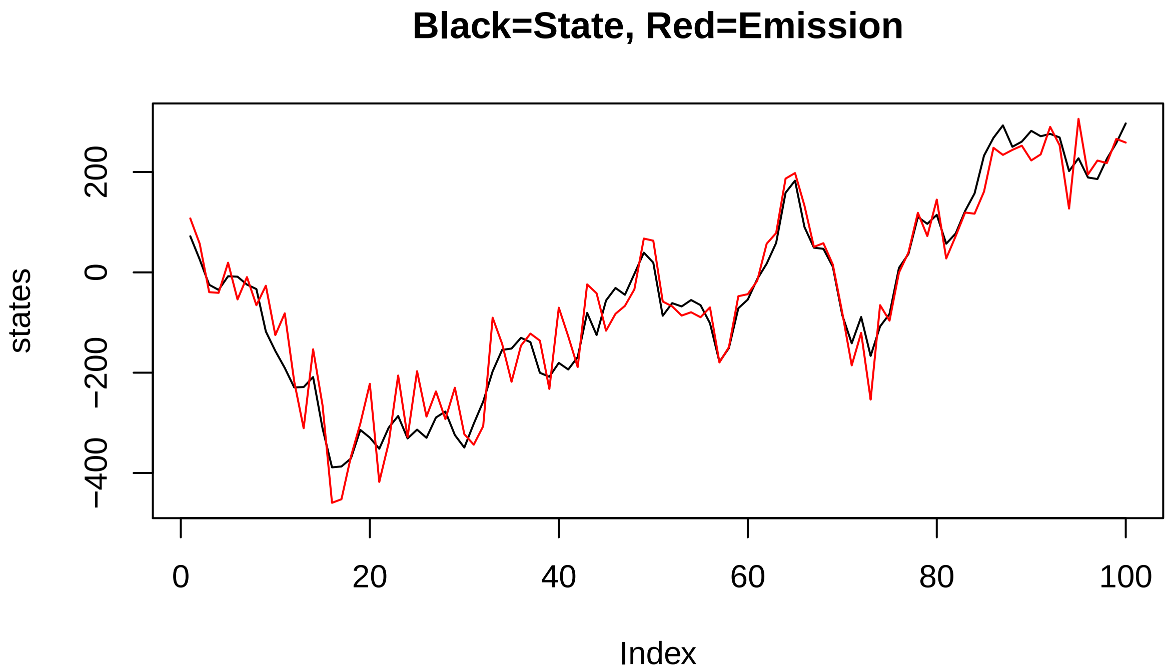

Black=State, Orange=Expected
states
40
80
120
160
0
20
40
60
80
100
1:nparticles

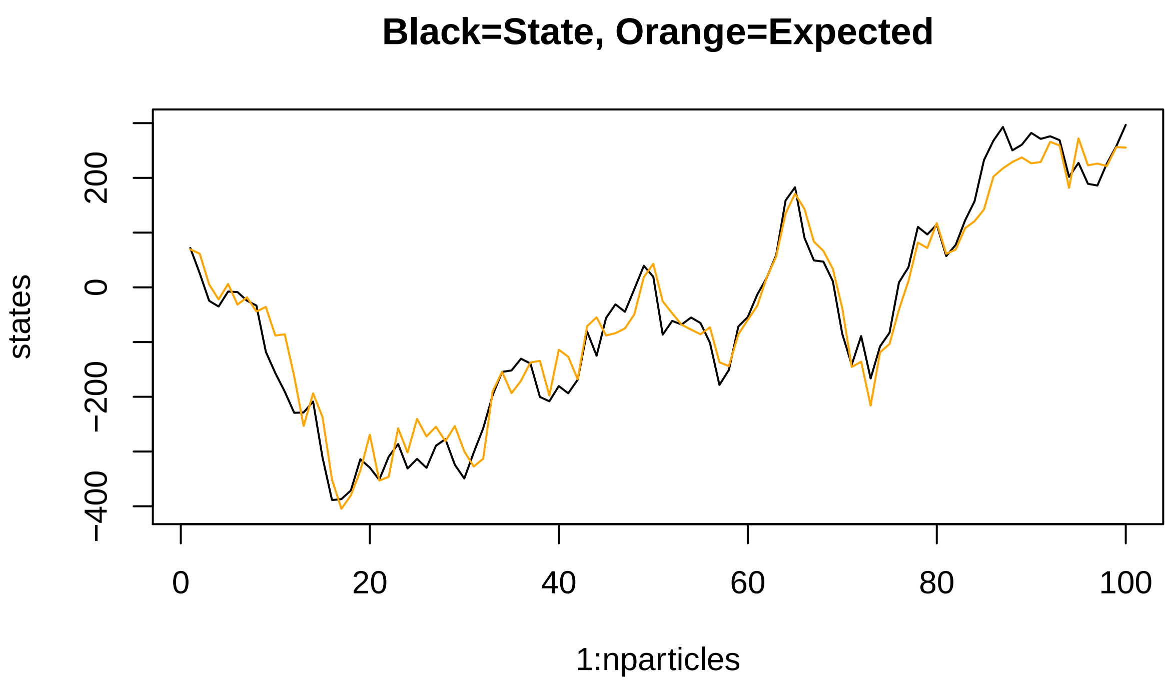


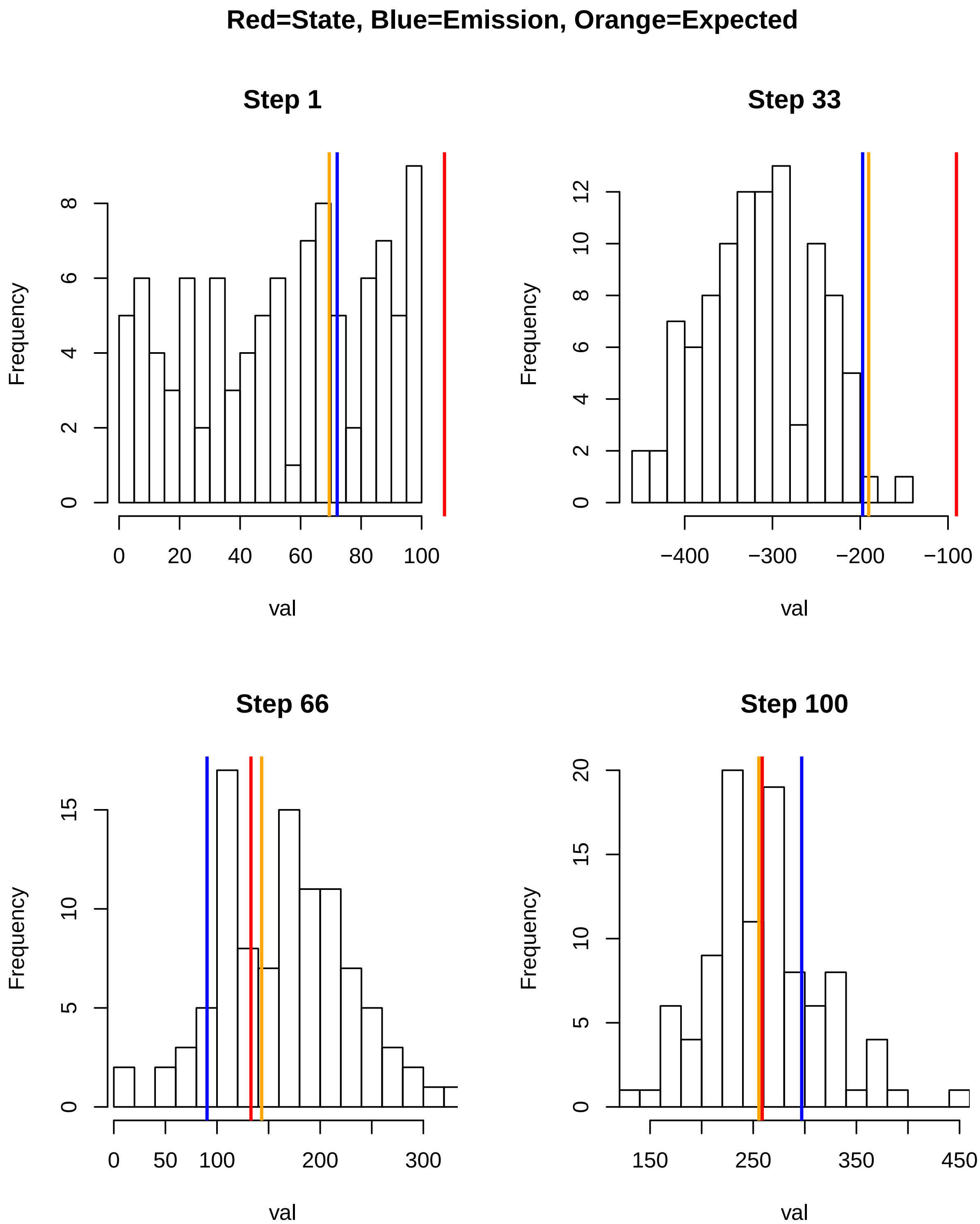
Red=State, Blue=Emission, Orange=Expected
Step 1
Frequency
0
2
4
6
8
0
20
40
60
80
100
val
Step 33
Frequency
0
5
10
15
40
45
50
55
60
65
70
val
Step 66
Frequency
0
5
10
15
120
130
140
150
val
Step 100
Frequency
0
5
10
15
165
175
185
val

Sigma 50

```r
sigma <- 50

set.seed(12345)
samples <- generate_states_and_emissions(100, sigma)

nparticles <- 100
particles <- particle_filtering(samples$emissions, nparticles, sigma)

expected <- get_expected_states(particles$particles, particles$weights)
```


Black=State, Red=Emission
states
200
0
−200
−400
0
20
40
60
80
100
Index

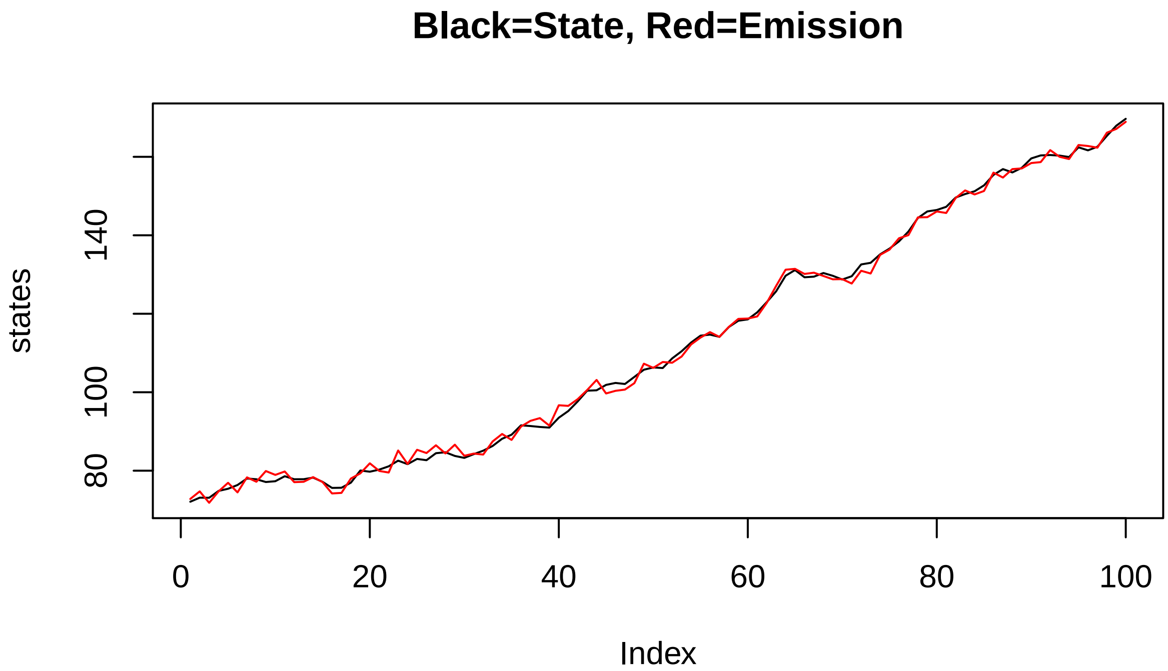

Black=State, Orange=Expected
states
200
0
−200
−400
0
20
40
60
80
100
1:nparticles

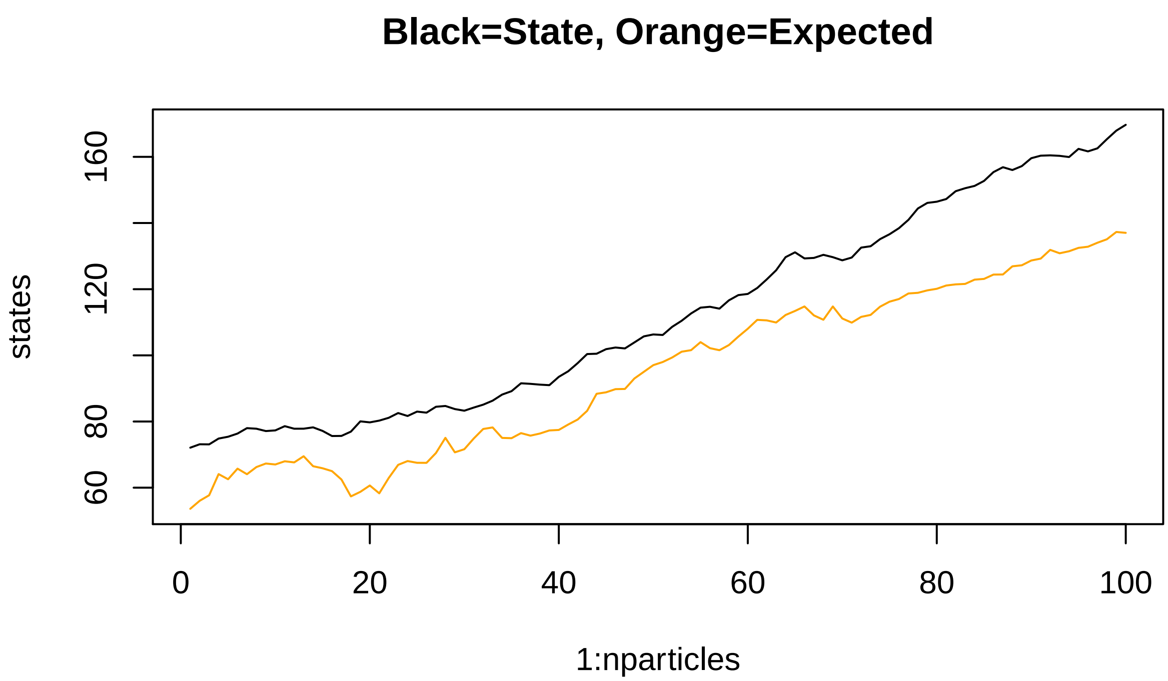

Red=State, Blue=Emission, Orange=Expected
Step 1
Frequency
0
2
4
6
8
0
20
40
60
80
100
val
Step 33
Frequency
0
2
4
6
8
10
12
−400
−300
−200
−100
val
Step 66
Frequency
0
5
10
15
0
50
100
200
300
val
Step 100
Frequency
0
5
10
15
20
150
250
350
450
val

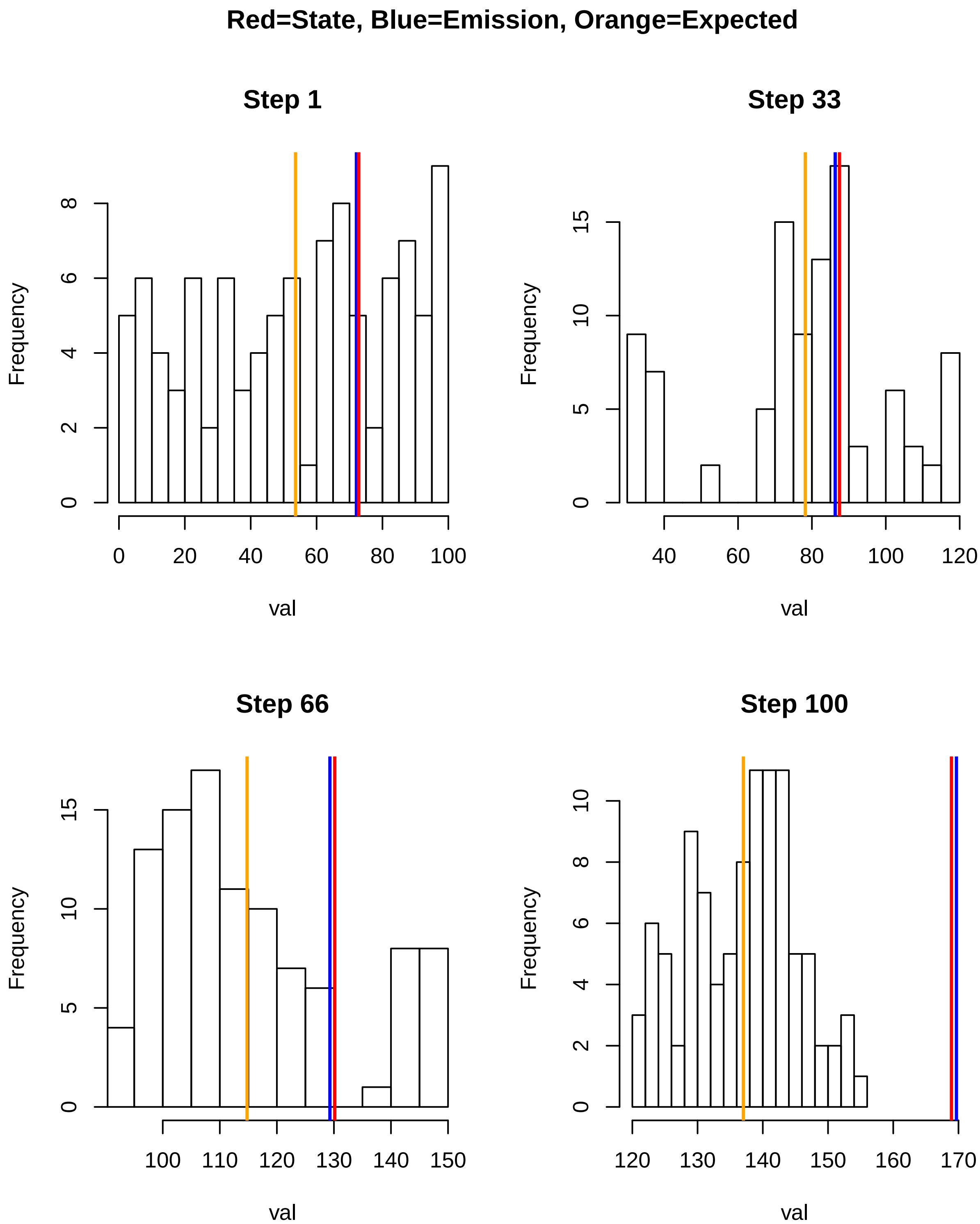
Sigma 1 without weight correction

```
get_uniform_sampling_weights <- function(emission, particles) {
    rep(1, length(particles)) / length(particles)
}

particle_filtering_without_correction <- function(x, nparticles, sigma) {
    steps <- length(x)
    particles <- matrix(NA, nrow=steps, ncol=nparticles)
    weights <- matrix(NA, nrow=steps, ncol=nparticles)

    particles[1, ] <- runif(nparticles, 0, 100)
    weights[1, ] <- get_uniform_sampling_weights(x[1], particles[1, ])

    for (i in 2:steps) {
        predictions <- sample(particles[i - 1, ], size=nparticles,
                              replace=TRUE, prob=weights[i - 1, ])
        particles[i, ] <- sapply(predictions, sample_transition, sigma=sigma)
        weights[i, ] <- get_uniform_sampling_weights(x[i], particles[i, ])
    }

    list(particles=particles, weights=weights)
}
```

```
sigma <- 1

set.seed(12345)
samples <- generate_states_and_emissions(100, sigma)

nparticles <- 100
particles <- particle_filtering_without_correction(samples$emissions, nparticles, sigma)

expected <- get_expected_states(particles$particles, particles$weights)
```


Black=State, Red=Emission
states
80
100
140
0
20
40
60
80
100
Index


Black=State, Orange=Expected
states
60
80
120
160
0
20
40
60
80
100
1:nparticles



Without using weight correction we do not use our emission data to learn better estimations of the true states. All particles becomes equally good no matter how likely or unlikely they correspond to the sensor readings.