Lab 4

Haochi Kiang (haoki222)

```
## Loading required package: RColorBrewer
```

Implementation

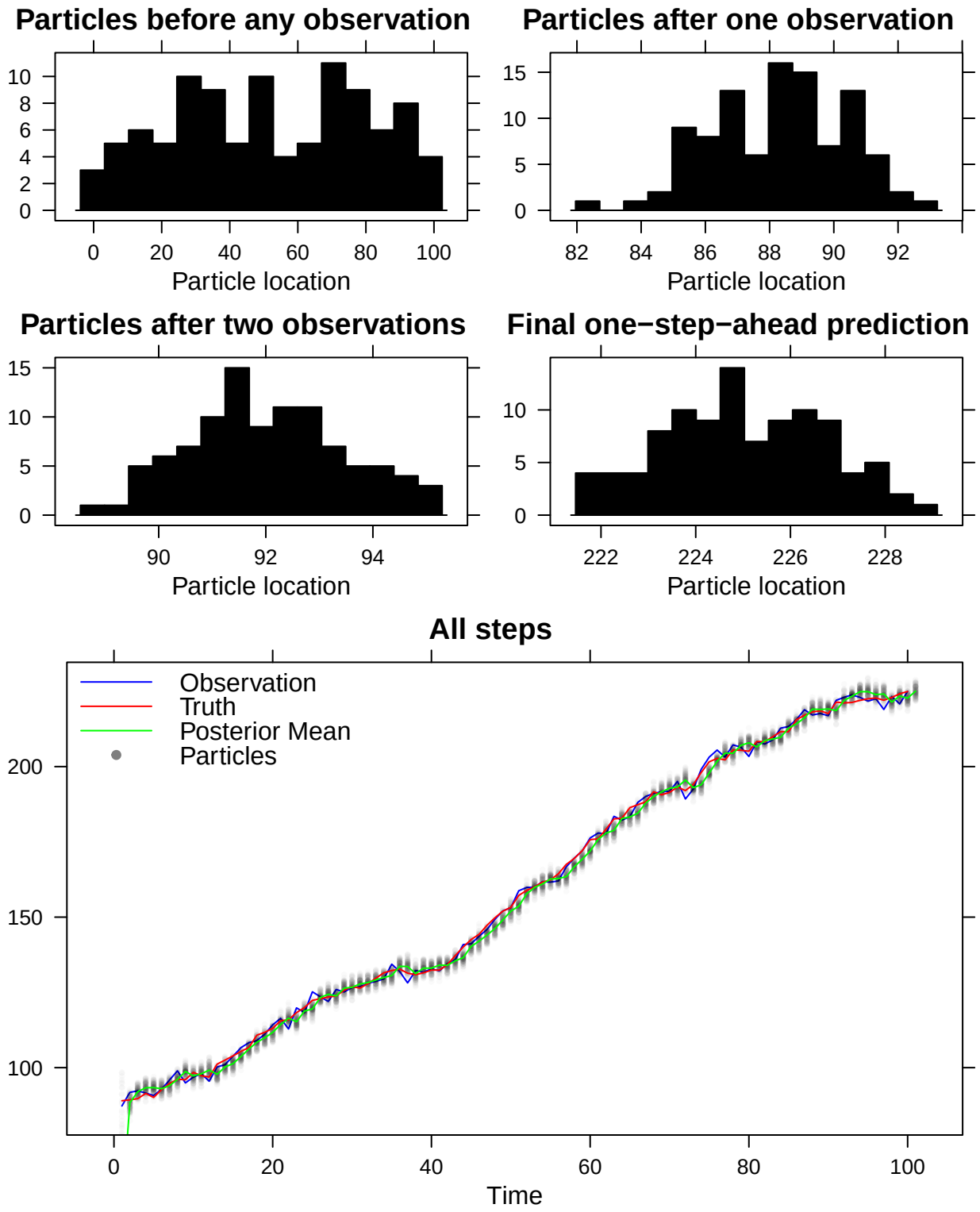
For implementation of the algorithm please see Appendix.

Emission standard deviation = 1

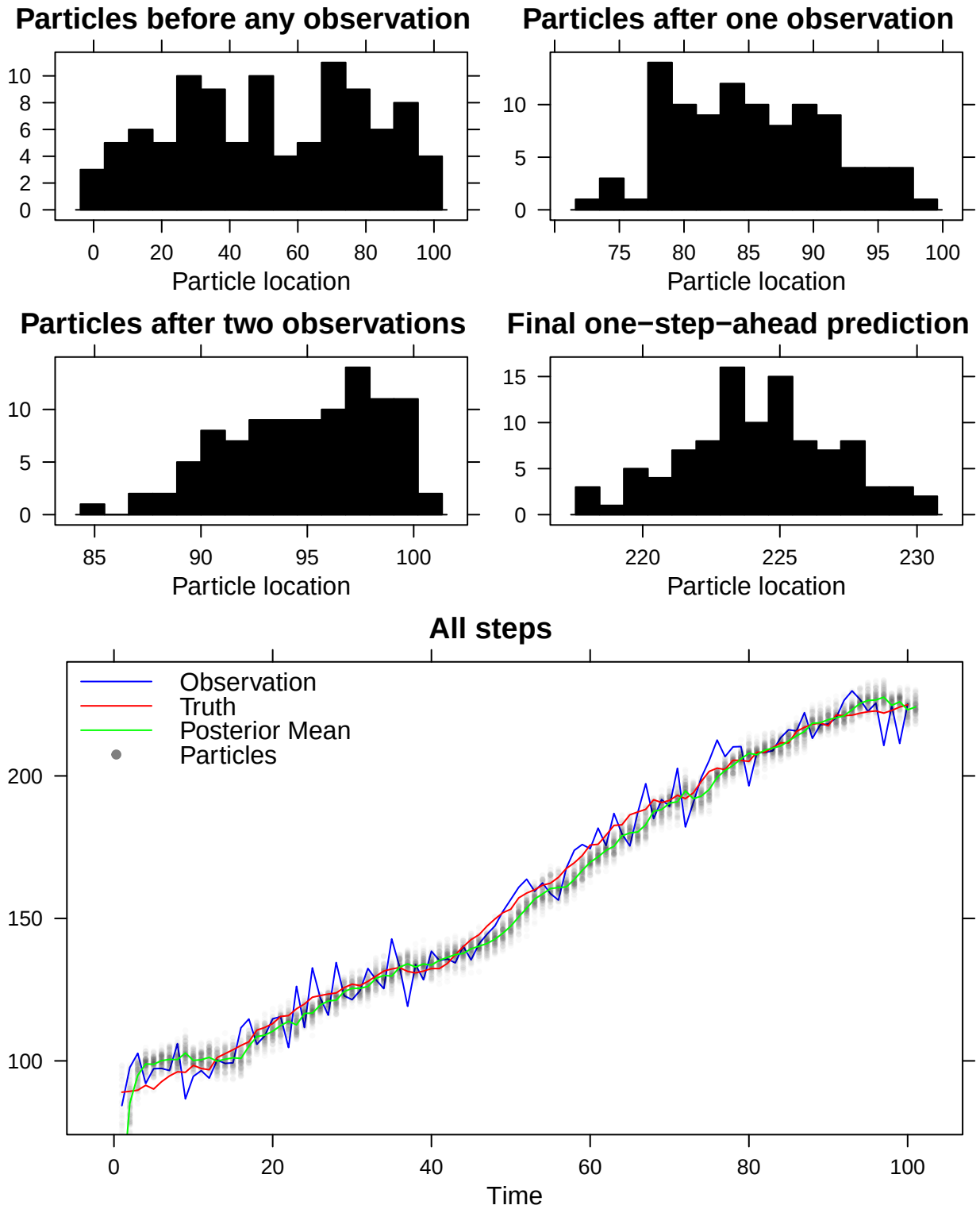
The following plot shows the particles at the first, second, third and the last step, as well as an overview of the entire process in a line plot. Emission standard deviation is set to 1 in this section.

The histogram of the particles at the beginning shows the state of the particles before observing any data points. In other words, according to the given model specification, this is uniform distribution with range between zero and one hundred.

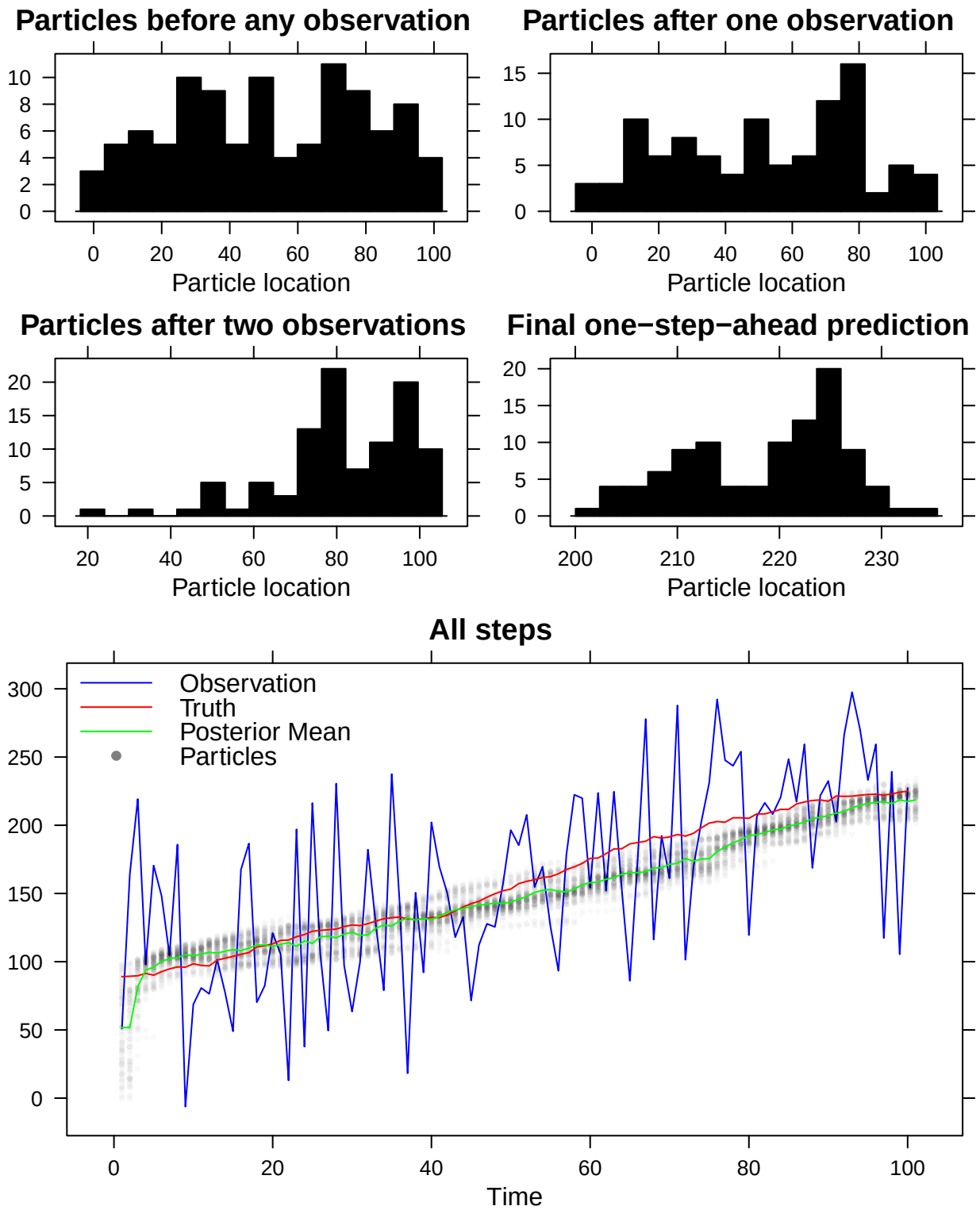
We can dr


Particles before any observation
Particle location
Particles after one observation
Particle location
Particles after two observations
Particle location
Final one−step−ahead prediction
Particle location
All steps
Observation
Truth
Posterior Mean
Particles
Time

Emission standard deviation = 5

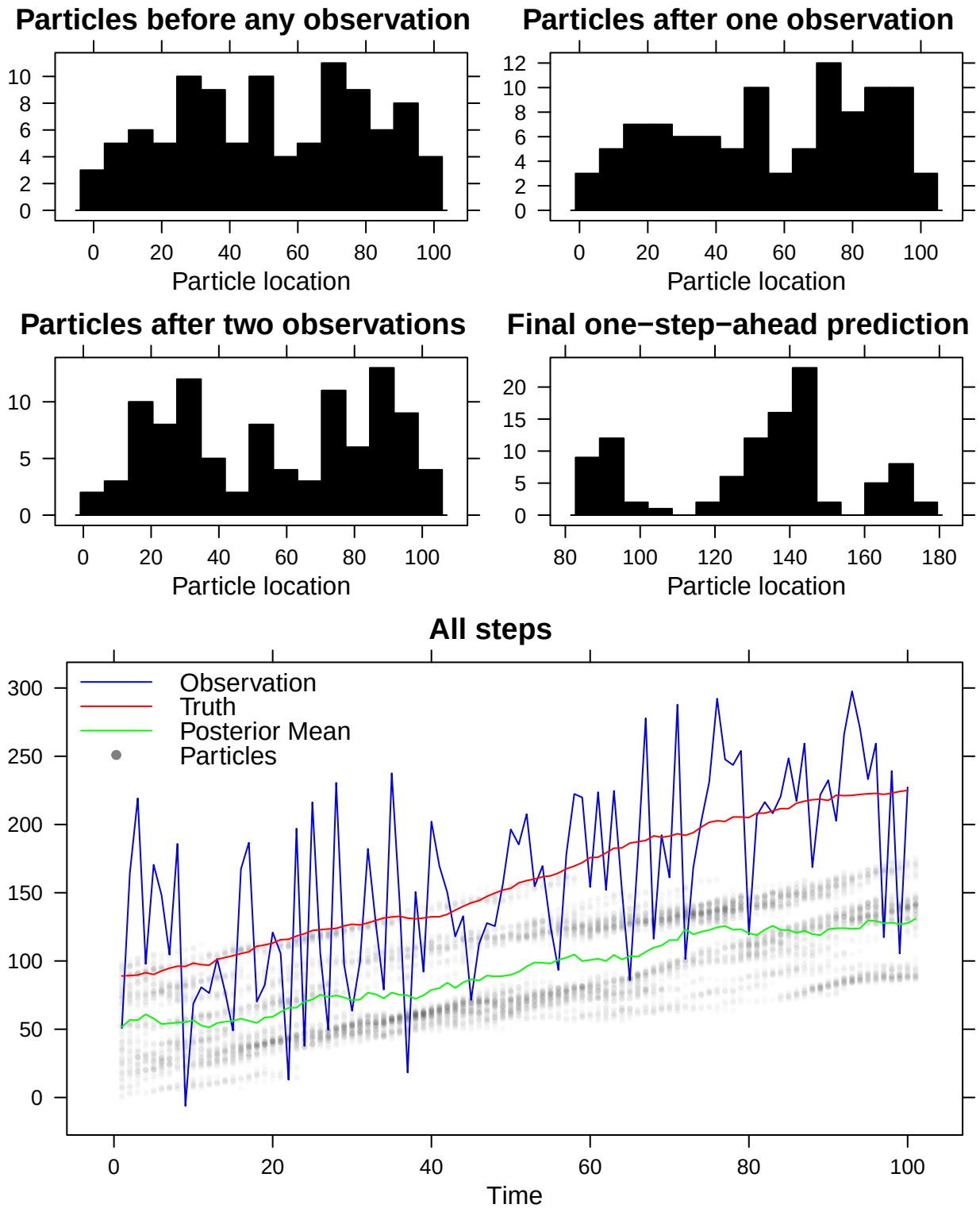


Emission standard deviation = 50



Always-constant weight, emission

```
set.seed(1111)
sim_and_plot(50, ahead_measr = ahead_measure_nowgt)
```


Particles before any observation
Particle location
Particles after one observation
Particle location
Particles after two observations
Particle location
Final one−step−ahead prediction
Particle location
All steps
Observation
Truth
Posterior Mean
Particles
Time

All codes

```
knitr::opts_chunk$set(fig.width=6.5, fig.height=8)

library(grid)
library(gridExtra)
library('lattice')
library('latticeExtra')
library('functional')
library('matrixStats') ## For numerically stable logSumExp

sumnorm = function (x) { print(x); x / sum(x) }

## Returns the log emission probability
demit = Vectorize(function(x, z, emit_sd) {
    lndens = dnorm(x, z + (-1:1), emit_sd, log = T)
    logSumExp(lndens) - log(3)
}, vectorize.args = 'z')

rtrans = Vectorize(function(z) {
    rnorm(1, z + (0:2)[sample(3,1)], 1)
})

sim_myssm = function(duration, emit_sd) {
    res = list()
    res$z = Reduce( function (lastz, mv) {
        c(lastz, rnorm(1, tail(lastz,1) + (0:2)[mv], 1))
    }, replicate(duration-1, sample(3,1)), init=runif(1,0,100))

    res$x = sapply( res$z, function (z) {
        rnorm(1, z + (-1:1)[sample(3,1)], emit_sd)
    })
    res
}

ahead_measure = function (cur_x, prev_measr, demitfn) {
    ## We don't need to normalize it here because sample() behaves the same
    ln_wgt = demitfn(cur_x, prev_measr)
    wgt = exp(ln_wgt - max(ln_wgt) + 10)
    rtrans(prev_measr[replicate(length(prev_measr),
                                sample(length(wgt), 1, replace=T, prob=wgt))])
}

ahead_measure_nowgt = function (cur_x, prev_measr, demitfn) {
    ## We don't need to normalize it here because sample() behaves the same
    wgt = rep(1, length(prev_measr))
    rtrans(prev_measr[replicate(length(prev_measr),
                                sample(length(wgt), 1, replace=T, prob=wgt))])
}


plot_history = function (measr_history, x, z) {
    timemax = ncol(measr_history)
    nsamp = nrow(measr_history)
    xy = do.call(cbind, lapply(1:timemax, function (tm)
        sapply(1:nsamp, function (s) c(tm, measr_history[s,tm]))))
    allplot = xyplot(x ~ seq_along(x), type='l', col=4, ylab='',
                     xlab = 'Time', main = 'All steps',
                     key = list(corner = c(0,0.98),
                                lines = list(col = c('blue','red','green', rgb(0,0,0,0.5)),
                                             type = c('l','l','l','p'), pch = 20),
                                text = list(c('Observation', 'Truth', 'Posterior Mean',
```

```
                                                'Particles')))) +
        as.layer(xyplot( xy[2,] ~ xy[1,], col=rgb(0,0,0,0.02),
                         pch = 20, cex = 0.5)) +
        as.layer(xyplot(z ~ seq_along(z), type='l', col=2)) +
        as.layer(xyplot( colMeans(measr_history) ~ 1:ncol(measr_history),
                         type='l', col='green'))
    beginplot = histogram(measr_history[,1], nint = 15, col = 1, ylab='',
                          main = 'Particles before any observation',
                          xlab = 'Particle location')
    secondplot = histogram(measr_history[,2], nint = 15, col = 1, ylab='',
                           main = 'Particles after one observation',
                           xlab = 'Particle location')
    thirdplot = histogram(measr_history[,3], nint = 15, col = 1, ylab='',
                          main = 'Particles after two observations',
                          xlab = 'Particle location')
    endplot = histogram(measr_history[,ncol(measr_history)],
                        nint = 15, col=1, ylab = '',
                        main='Final one-step-ahead prediction',
                        xlab = 'Particle location')

    grid.arrange(
        grobs = list(beginplot, secondplot,
                     thirdplot, endplot,
                     allplot),
        layout_matrix = rbind(c(1,2),
                              c(3,4),
                              c(5,5),
                              c(5,5)))
}

sim_and_plot = function (emit_sd, nsamp=100, ahead_measr = ahead_measure) {
    demit1 = Curry(demit, emit_sd = emit_sd)
    modsamp1 = sim_myssm(100, emit_sd)
    measr_history = matrix(NA, nsamp, length(modsamp1$x)+1)
    measr_history[,1] = runif(nsamp, 0, 100)
    for (i in seq_len(length(modsamp1$x)) ) {
        measr_history[,i+1] =
            ahead_measr(modsamp1$x[i], measr_history[,i], demit1)
    }
    plot_history(measr_history, modsamp1$x, modsamp1$z)

}

lattice.options(
    layout.heights=list(bottom.padding=list(x=0),
                        top.padding=list(x=0.4),
                        main.key.padding = list(x=-1),
                        axis.xlab.padding = list(x=0)),
  layout.widths=list(left.padding=list(x=-0.5), right.padding=list(x=-1))
)

set.seed(1111)
sim_and_plot(1)

set.seed(1111)
sim_and_plot(5)

set.seed(1111)
sim_and_plot(50)

set.seed(1111)
sim_and_plot(50, ahead_measr = ahead_measure_nowgt)
```