```r
library('HMM')
library('lattice')
library('latticeExtra')
```

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

```r
library('entropy')
library('doParallel')
```

```
## Loading required package: foreach

## Loading required package: iterators

## Loading required package: parallel
```

```r
library('foreach')
library('vioplot')
```

```
## Loading required package: sm

## Package 'sm', version 2.2-5.4: type help(sm) for summary information
```

```r
library('functional')
```

```r
ringsize = 10
accur = 2
emit_p = sapply(1:ringsize, function (st) {
    p = rep(0, ringsize)
    p[((st-1-accur):(st-1+accur)) %% ringsize + 1] = 1/(2*accur+1)
    p
})
trans_p = sapply(1:ringsize, function (st) {
    p = rep(0, ringsize)
    p[((st-1-1):(st-1+1)) %% ringsize + 1] = 1/3
    p
})
hmmmod = initHMM(1:ringsize, 1:ringsize, transProbs = trans_p, emissionProbs = emit_p)

robotsamp1 = simHMM(hmmmod, 100)
```

```r
forwardp = forward(hmmmod, robotsamp1$observation)
posteriorp = posterior(hmmmod, robotsamp1$observation)

map_path = viterbi(hmmmod, robotsamp1$observation)
forward_pred = apply(forwardp, 2, which.max)
posterior_pred = apply(posteriorp, 2, which.max)
```

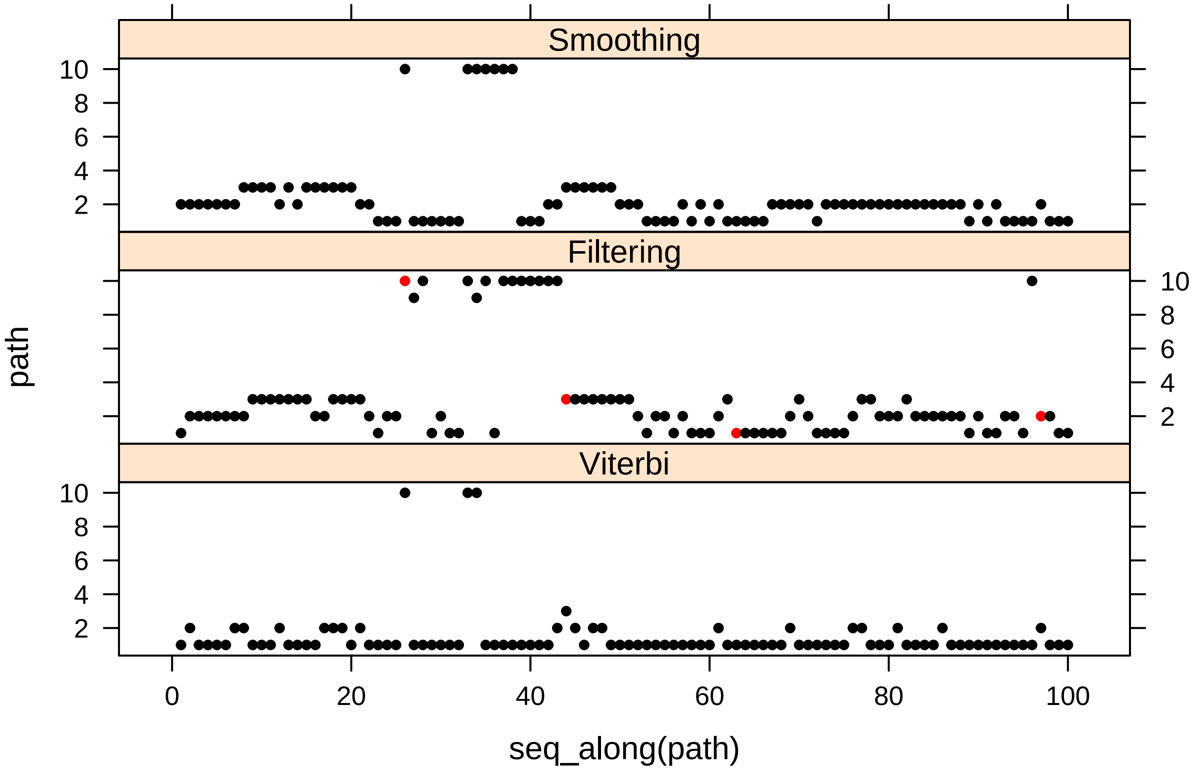
```r
## Compute element-wise clock-distance on the monogenous group of order `p`
## `x` and `y` should be coded 1:p
monogroup_dist = function (x, y, p)
    pmin(abs((x-1) %% p - y+1), 10 - abs((x-1) %% p - y+1))

plot_path = function (path) {
    ## Highlight impossible moves
    dif = (monogroup_dist(path[1:(length(path)-1)], path[2:length(path)], ringsize) > 1) + 1
    xyplot( path ~ seq_along(path), col = c(1,dif), pch = 20 )
}
```

```
c('Viterbi' = plot_path(map_path),
  'Filtering' = plot_path(forward_pred),
  'Smoothing' = plot_path(posterior_pred),
  layout = c(1,3))
```

```
## Warning in formals(fun): argument is not a function

## Warning in formals(fun): argument is not a function
```



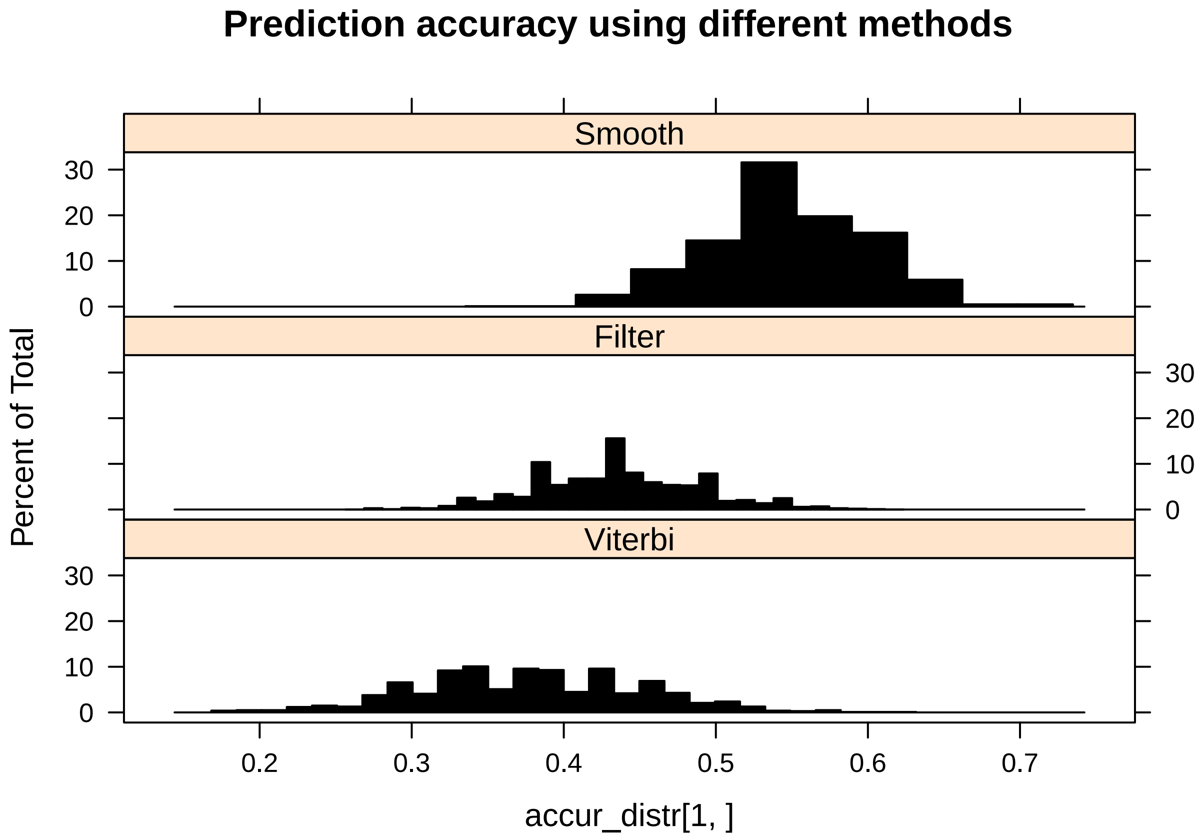
```
accur = function (truth, pred) {
    sum(pred == truth) / length(pred)
}
prediction_accuracies = c('Viterbi' = accur(robotsamp1$states, map_path),
                          'Filter' = accur(robotsamp1$states, forward_pred),
                          'Smooth' = accur(robotsamp1$states, posterior_pred))
```

```
accur_distr = replicate( 1000, {
    s = simHMM(hmmmod, 100)
    forwardp = forward(hmmmod, s$observation)
    posteriorp = posterior(hmmmod, s$observation)
    map_path = viterbi(hmmmod, s$observation)
    forward_pred = apply(forwardp, 2, which.max)
    posterior_pred = apply(posteriorp, 2, which.max)
    c('Viterbi' = accur(s$states, map_path),
      'Filter' = accur(s$states, forward_pred),
      'Smooth' = accur(s$states, posterior_pred))
})
```

```
c(Viterbi = histogram(~accur_distr[1,], nint = 30, col = 1,
                      main = 'Prediction accuracy using different methods'),
  Filter = histogram(~accur_distr[2,], nint = 30, col = 1),
  Smooth = histogram(~accur_distr[3,], init = 30, col = 1),
  x.same = T, y.same = T, layout = c(1,3))
```

```
## Warning in formals(fun): argument is not a function

## Warning in formals(fun): argument is not a function
```

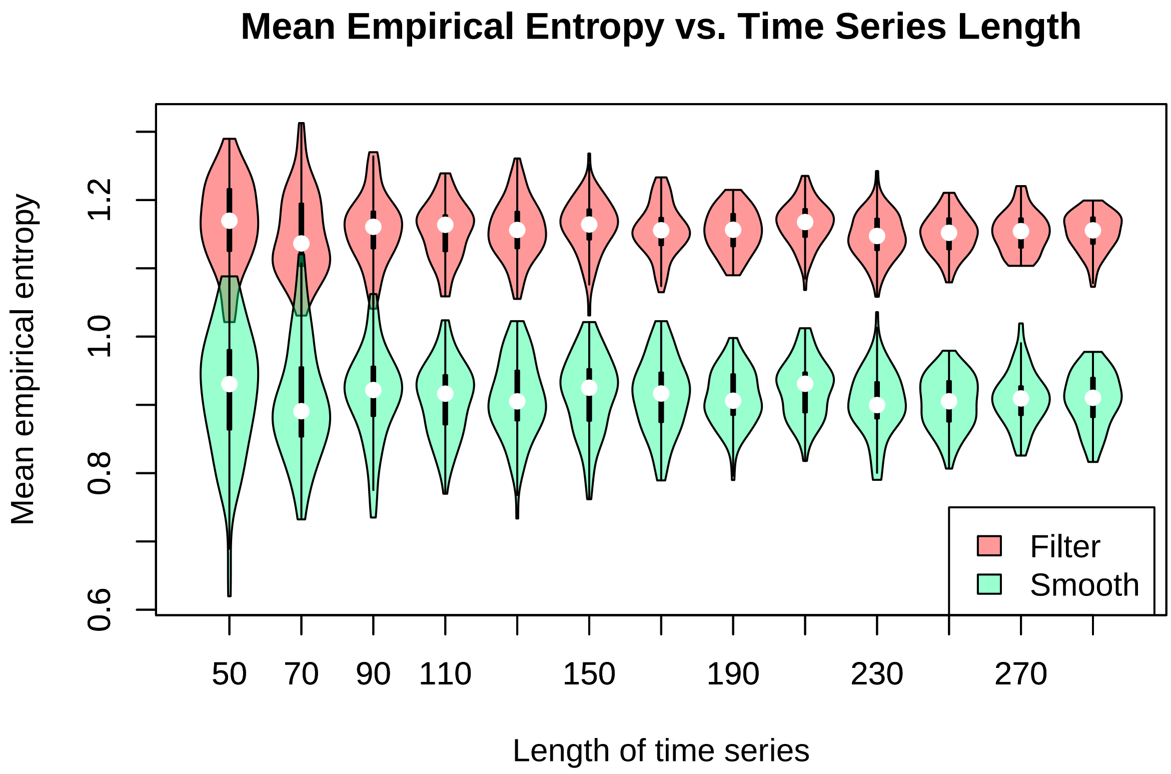


```
registerDoParallel(cores=8)
ent = foreach (simlen = seq(50, 290, by = 20)) %dopar% {
    cat('Doing simlen =', simlen, '\n')
    replicate(80, {
        s = simHMM(hmmmod, simlen)
        forwardp = forward(hmmmod, s$observation)
        posteriorp = posterior(hmmmod, s$observation)
        c('Filter' = mean(apply( prop.table(exp(forwardp),2), 2, entropy.empirical)),
          'Smooth' = mean(apply( posteriorp, 2, entropy.empirical)))
    })
}

ent = simplify2array(ent)
dimnames(ent)[[3]] = seq(50, 290, by = 20)
```

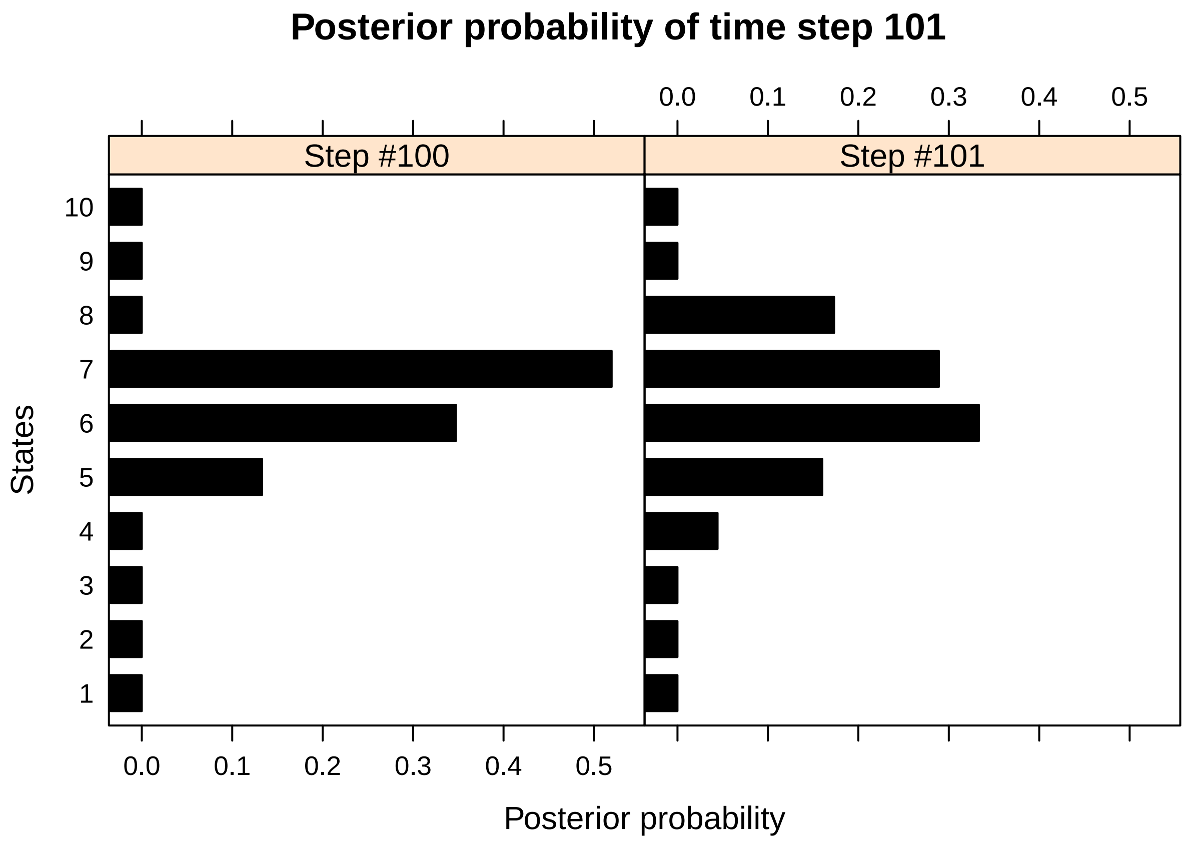
```
vplot = function (x, ...) {
    do.call( vioplot, c(lapply(1:ncol(x), function (i) tmp = x[,i]), add = T, ...) )
    axis(side=1, at=seq(ncol(x)), labels=dimnames(x)[[2]])
    axis(side=2)
}

plot(0:1, 0:1, type='n', xlim=c(0.5,13+0.5),
     ylim = range(ent),
     axes=FALSE,ann=FALSE)
vplot(ent[1,,], col = '#FF000066')
vplot(ent[2,,], col = '#00FF8866')
title(main = 'Mean Empirical Entropy vs. Time Series Length',
      xlab = 'Length of time series', ylab = 'Mean empirical entropy');
legend(11, 0.75, legend = c('Filter','Smooth'),
       fill = c('#FF000066','#00FF8866'))
```



```
p101 = rowSums(sapply(1:nrow(posteriorp), function(st) {
    posteriorp[st,100] * trans_p[st,]
}))
c('Step #100' = barchart(1:10 ~ posteriorp[,100], col = 1,
                         xlab = 'Posterior probability',
                         ylab = 'States',
                         main = 'Posterior probability of time step 101'),
  'Step #101' = barchart(1:10 ~ p101, col = 1),
  x.same = T,
  y.same = T)
```

```
## Warning in formals(fun): argument is not a function
```



```
robotsamp1$states[100]
```

```
## [1] 1
```