Posterior and EM

Camrin Braun

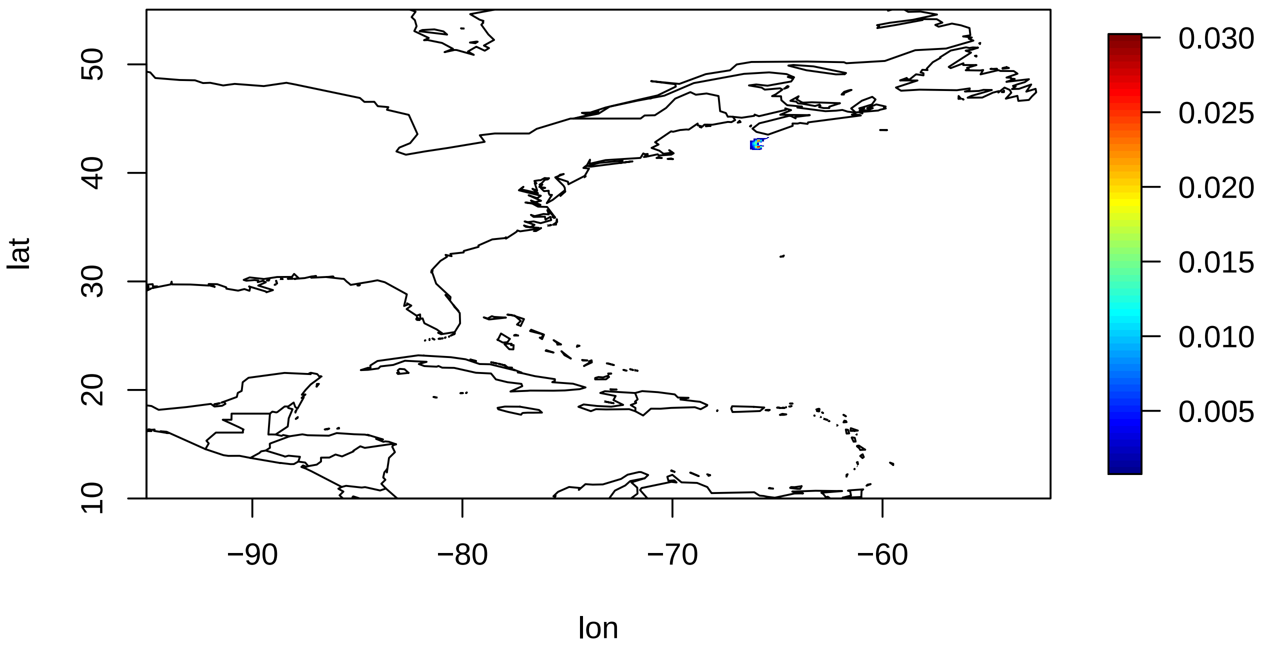
October 11, 2016

Posterior distribution

From Pedersen et al 2011: *When fitting an SSM in a Bayesian context the smoothing step provides the posterior distribution of the state…it is therefore common to use time marginals of the posterior distribution (i.e. probability distributions of all states at specific times) which, in fact, are the state estimates returned from the HMM smoothing algorithm.*

For us, this is the output of `hmm.smooth` that we call s of dim(states, time, x, y). In a classic EM approach (non-stochastic) the E-step is simply the computation of the posterior distribution given current model parameter estimates which we've already done in s.

```
##  num [1:2, 1:135, 1:539, 1:564] 0 0 0 0 0 0 0 0 0 0 ...
```



So the E-step in a traditional EM sense should be complete. If we want to go the stochastic route as in Woillez then we need to finish the E-step with sampling of N trajectories (which I honestly don't really understand but seems like it would be straightforward).

The M-step "updates model parameters according to a ML criterion reweighted by the posterior distribution."(from Woillez) I don't understand how this works either and, if my assumptions are correct, this is where we need the help from Andy? We need to update D and P(switch) before re-running kernels, filter and smoother. Something like:

```
# GENERATE MOVEMENT KERNELS. D VALUES ARE MEAN AND SD PIXELS
K1 = as.cimg(gausskern(D1[1], D1[2], muadv = 0))
K2 = as.cimg(gausskern(D2[1], D2[2], muadv = 0))
P <- matrix(c(p[1],1-p[1],1-p[2],p[2]),2,2,byrow=TRUE)
```

```r
# RUN THE FILTER STEP
f = hmm.filter(g,L,K1,K2,P)

# RUN THE SMOOTHING STEP
s = hmm.smoother(f, K1, K2, P, plot = F)

# UPDATE D


# UPDATE P
# get states from s
sv <- apply(s[1,,,], 1, sum) > apply(s[2,,,], 1, sum)
sv <- -sv + 2

# difference it to find transitions
svd = rbind(sv, c(0,diff(sv)))

# get a new transition probability matrix
r1 = rev(table(svd[2, svd[1,]==1])/sum(svd[1,]==1))
r2 = rev(table(svd[2, svd[1,]==2])/sum(svd[1,]==2))
P = matrix(rbind(r1, r2), 2,2)

# CALCULATE CHANGE IN PARAMETER VALUES OVER LAST SEVERAL ITERATIONS
# Woillez used "ratio betwen the average over the last 20 values of D and the new value of D below 1%"


# RE-ITERATE
# now have a new set of diffusion parameters (D1, D2) and switching probability (P) and we can iterate
```