Testing dpf

Daniel J. McDonald

10/23/2017

Testing

I tried doing the following:

```
# pull recent mtmcbride/dpf/dpf
# devtools::install_github('mtmcbride/dpf/dpf')
library(dpf)
```

which had some problems that I'm not quite sure of. I managed to download it, open the file structure and build using `devtools::load_all()`.

```
##devtools::load_all('../dpf')
## Note that this is my forked version.
```

I then made a few modifications for testing to `dpf.cpp`. These are incorporated in my fork. (Which has the same issue). In `kf1step` I used

```
arma::mat a1 = a0;
arma::mat pred = ct + Zt * a0; // added this between 152 and 153
arma::mat vt = yt - pred;
arma::mat Ft = GGt + Zt * P0 * Zt.t();
arma::mat Ftinv = arma::inv(Ft);
arma::mat Kt = P0 * Zt.t() * Ftinv;
a1 += Kt * vt;
```

and changed the output to

```
return List::create(Named("a1") = a1,
                    Named("P1") = P1,
                    Named("lik") = lik,
                    Named("pred") = pred); // added this line 172
```

These two changes should allow me to see the predictions (the continuous states you actually want to plot) for the *previous* state.

Second, I changed the following in `pathStuff`:

```
means(iter) = step["pred"];
arma::mat aa0 = step["a1"];
arma::mat PP0 = step["P1"];
double liktmp = step["lik"];
llik(iter) += log(liktmp);
```

And made it return a list:

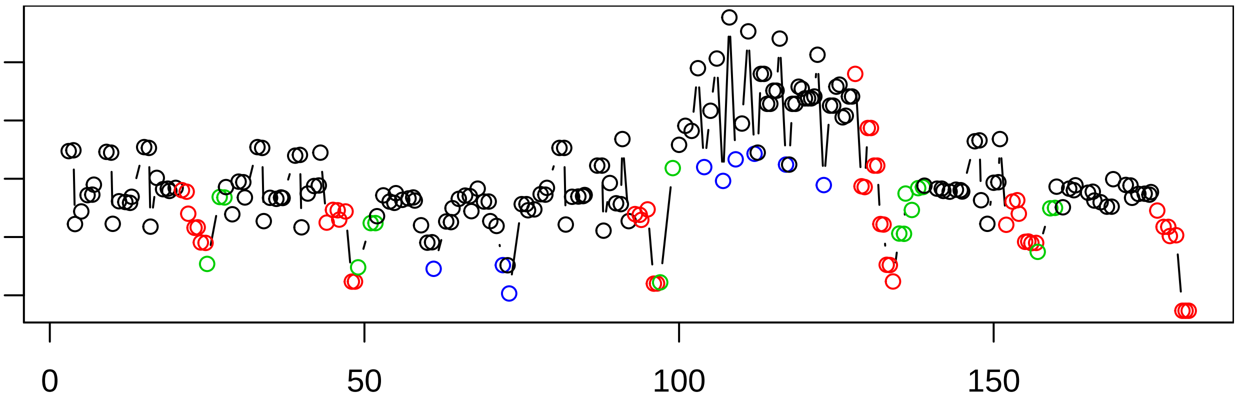
```
return List::create(Named("preds") = means,
                      Named("llik") = llik);
```

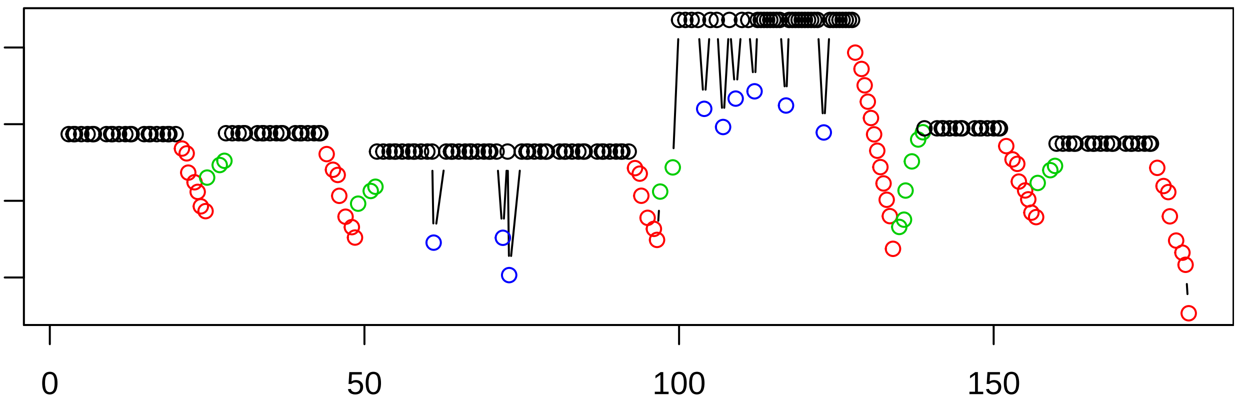
This should let me look at the predictions easily if I get a path.

Finally, I removed the `temposwitch` argument from `yupengMats` and reduced the number of *mu* parameters to 3.

At this point, I get the same issues that you had (all the predictions are the same).

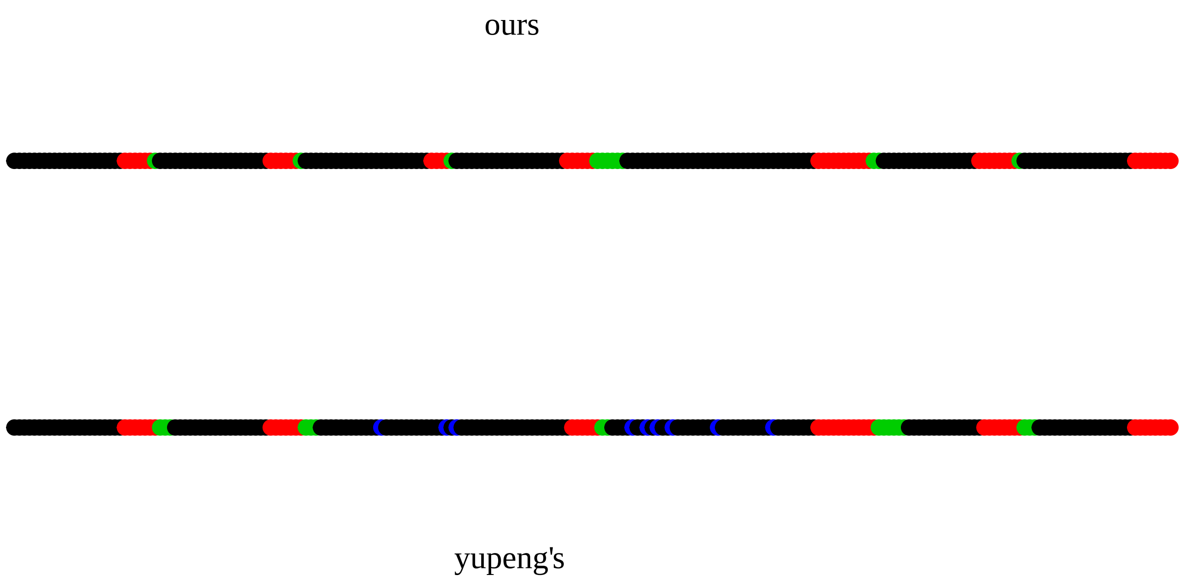
```
source('../Beam_Search.R') ## Yupeng's original beam search on some music data
```





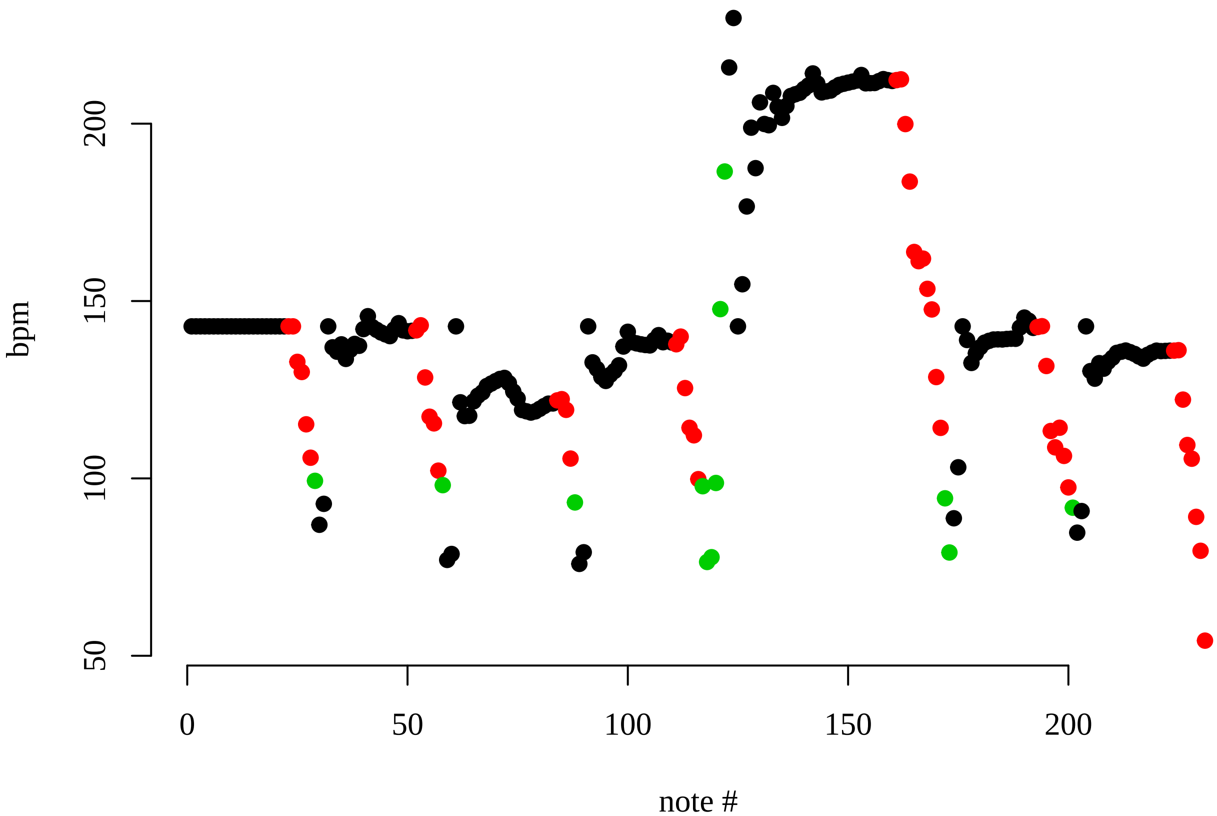
This plots our discrete states and his.

```
lt = ioi
out = yupengMats(lt, sd_l^2, c(mean(y),-40,-40), c(.01,20,10,30)^2, c(.8,.1,.8,.4))
test = beamSearch(out$a0,out$P0,c(1,0,0,0,0,0,0,0), out$dt, out$ct, out$Tt, out$Zt,
out$Rt, out$Qt, out$GGt, matrix(y,nrow = 1), out$transMat, 200)
source('../../R/stateConversion.R')
nn = length(y)
bestpath = test$paths[which.max(test$weights),]
par(mfrow=c(1,1))
par(mar=c(0,0,0,0),family='serif')
plot(1:nn,rep(1,nn),col=convert8to4(bestpath),
     pch=19,ylim=c(-1,2),bty='n',xlab='',ylab='',yaxt='n',xaxt='n')
points(1:nn, rep(0,nn), col=s, pch=19)
text(100,1.5,"ours")
text(100,-.5,"yupeng's")
```



This plots our continuous states.

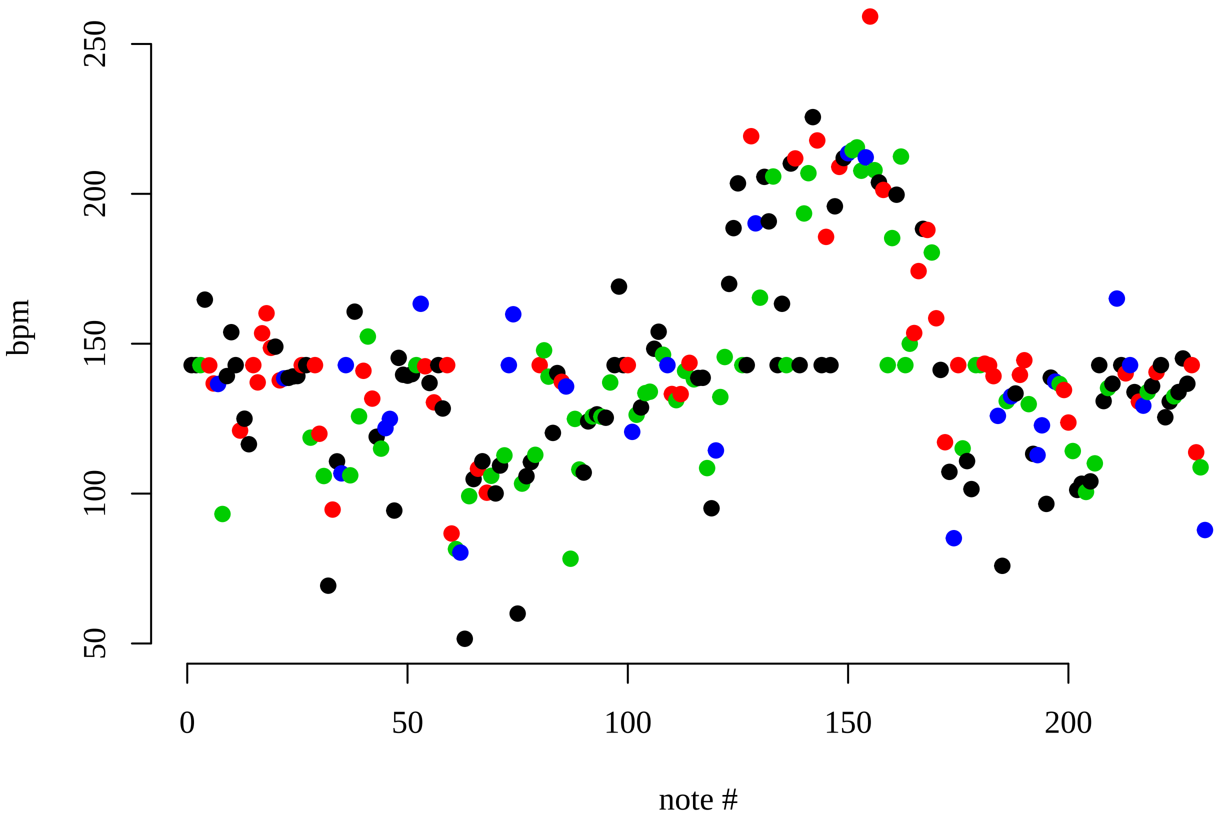
```
filt = pathStuff(out, bestpath, matrix(y,nrow=1))
par(mar=c(5,4,0,0),family='serif')
plot(1:nn,filt$preds[1:nn],col=convert8to4(bestpath),
     pch=19,bty='n',xlab='note #',ylab='bpm')
```



Here is a plot of our continuous states for a **random** path:

```
set.seed(12345)
rands = sample(0:7, nn, replace=TRUE)
```

```r
randfilt = pathStuff(out, rands, matrix(y,nrow=1))
par(mar=c(5,4,0,0),family='serif')
plot(1:nn,randfilt$preds[1:nn],col=convert8to4(rands),
     pch=19,bty='n',xlab='note #',ylab='bpm')
```



As you can see, these last two are identical. This is bad. So I wrote the `kf11` function to try to determine the issue. We start with `a0` and `P0` as before, and see what should be happening with the predictions if we iterate a few steps:

```r
(z1 = kf11(out, y, 1,1)) # the first 1 uses y[1],
```

```
## $a1
##          [,1]
## [1,] 142.8582
## [2,]   0.0000
##
## $P1
##       [,1]
## [1,] 2e-04
## [2,] 0e+00
## [3,] 0e+00
## [4,] 0e+00
##
## $lik
## [1] 0.006053142
##
## $pred
##          [,1]
```

```
## [1,] 142.8582
```

```r
    # the second is the state to trans to (base 1 index)
(z2 = kf11(within(out, {a0=z1$a1; P0=z1$P1}), y, 2, 4))
```

```
## $a1
##          [,1]
## [1,] 142.8582
## [2,]   0.0000
##
## $P1
##              [,1]
## [1,] 0.0002999999
## [2,] 0.0000000000
## [3,] 0.0000000000
## [4,] 0.0000000000
##
## $lik
## [1] 0.005743167
##
## $pred
##          [,1]
## [1,] 142.8582
```

```r
(z3 = kf11(within(out, {a0=z2$a1; P0=z2$P1}), y, 3, 4))
```

```
## $a1
##          [,1]
## [1,] 142.8582
## [2,]   0.0000
##
## $P1
##              [,1]
## [1,] 0.0003999997
## [2,] 0.0000000000
## [3,] 0.0000000000
## [4,] 0.0000000000
##
## $lik
## [1] 0.005658942
##
## $pred
##          [,1]
## [1,] 142.8582
```

```r
(z4 = kf11(within(out, {a0=z3$a1; P0=z3$P1}), y, 4, 5))
```

```
## $a1
##          [,1]
## [1,] 156.1915
## [2,]  40.0000
##
## $P1
##           [,1]
## [1,]  44.44494
## [2,] 133.33333
```

```
## [3,] 133.33333
## [4,] 400.00000
##
## $lik
## [1] 0.01155026
##
## $pred
##           [,1]
## [1,] 142.8582
```

```r
(z5 = kf11(within(out, {a0=z4$a1; P0=z4$P1}), y, 5, 1))
```

```
## $a1
##           [,1]
## [1,] 154.1675
## [2,]   0.0000
##
## $P1
##          [,1]
## [1,] 40.0005
## [2,]  0.0000
## [3,]  0.0000
## [4,]  0.0000
##
## $lik
## [1] 0.01193585
##
## $pred
##           [,1]
## [1,] 156.1915
```

I printed all of it, but it's clear that the predictions **should** be changing. Repeating the same thing and staying in the first state (as the best path suggests)...

```r
(z1 = kf11(out, y, 1,1)) # the first 1 uses y[1],
```

```
## $a1
##           [,1]
## [1,] 142.8582
## [2,]   0.0000
##
## $P1
##        [,1]
## [1,] 2e-04
## [2,] 0e+00
## [3,] 0e+00
## [4,] 0e+00
##
## $lik
## [1] 0.006053142
##
## $pred
##           [,1]
## [1,] 142.8582
```

```r
    # the second is the state to trans to (base 1 index)
(z2 = kf11(within(out, {a0=z1$a1; P0=z1$P1}), y, 2, 1))
```

```
## $a1
##          [,1]
## [1,] 142.8582
## [2,]   0.0000
##
## $P1
##              [,1]
## [1,] 0.0002999999
## [2,] 0.0000000000
## [3,] 0.0000000000
## [4,] 0.0000000000
##
## $lik
## [1] 0.005743167
##
## $pred
##          [,1]
## [1,] 142.8582
```

```r
(z3 = kf11(within(out, {a0=z2$a1; P0=z2$P1}), y, 3, 1))
```

```
## $a1
##          [,1]
## [1,] 142.8582
## [2,]   0.0000
##
## $P1
##              [,1]
## [1,] 0.0003999997
## [2,] 0.0000000000
## [3,] 0.0000000000
## [4,] 0.0000000000
##
## $lik
## [1] 0.005658942
##
## $pred
##          [,1]
## [1,] 142.8582
```

```r
(z4 = kf11(within(out, {a0=z3$a1; P0=z3$P1}), y, 4, 1))
```

```
## $a1
##          [,1]
## [1,] 142.8582
## [2,]   0.0000
##
## $P1
##              [,1]
## [1,] 0.0004999993
## [2,] 0.0000000000
## [3,] 0.0000000000
```

```
## [4,] 0.0000000000
##
## $lik
## [1] 0.01155026
##
## $pred
##          [,1]
## [1,] 142.8582
```

```
(z5 = kf11(within(out, {a0=z4$a1; P0=z4$P1}), y, 5, 1))
```

```
## $a1
##          [,1]
## [1,] 142.8582
## [2,]   0.0000
##
## $P1
##              [,1]
## [1,] 0.0005999986
## [2,] 0.0000000000
## [3,] 0.0000000000
## [4,] 0.0000000000
##
## $lik
## [1] 0.01879252
##
## $pred
##          [,1]
## [1,] 142.8582
```

So we need to know why are the predicted values not changing in `pathStuff`? (I doubt that they are correct in `beamSearch` either, so I imagine we are grabbing the wrong entries of something before calling `kf1step`.) Also, in `kf1step`, there is an error **(!!)**. It should have (line 158)

```
arma::mat P1 = P0 - P0 * Zt.t() * Kt.t();
```