Phylo-HMMs

M. J. Hubisz, K. S. Pollard, and A. Siepel

November 10, 2010

In this example we will use RPHAST to create a custom phylo-HMM for use in detecting transcription factor binding sites. As in the second example, we will make use of multiple alignments and a neutral phylogenetic model from UCSC. In addition, we will make use of a set of putative binding sites for the neuron-restrictive silencer factor (NRSF) in training the phylo-HMM. For convenience, we have included multiple alignments corresponding to these NRSF binding sites in the RPHAST package.

We begin by initializing RPHAST and loading the neutral model obtained from UCSC. For simplicity and speed, we will consider only a subset of the species in the model.

```
> require("rphast")
> seqnames <- c("hg18", "panTro2", "ponAbe2", "rheMac2", "equCab2",
+               "canFam2", "dasNov2", "mm9", "rn4", "cavPor3")
> exampleArchive <- system.file("extdata", "examples.zip", package="rphast")
> unzip(exampleArchive, "placentalMammals.mod")
> neutralMod <- read.tm("placentalMammals.mod")
> neutralMod$tree <- prune.tree(neutralMod$tree, seqs=seqnames, all.but=TRUE)
```

Now let's unpack the NRSF alignments and concatenate them into one large alignment. This will give us a convenient way of estimating a whole set of phylogenetic models—one per position in the motif—using a single RPHAST command.

There are a few details that need to be addressed as we read in these individual alignments (one per binding site). First, columns in these alignments that contain gaps in the reference genome must be discarded, so that the number of columns in each alignment equals the number of positions in the NRSF motif (21). In addition, alignments that correspond to binding sites on the negative strand must be reverse complemented.

```
> unzip(system.file("extdata", "NRSF.zip", package="rphast"))
> mafFiles <- list.files("NRSF", pattern="*.maf", full.names=TRUE)
> nrsfNames <- sub(".maf$", "", basename(mafFiles))  # remove dir and .maf
> nrsfSites <- read.feat("NRSF/NRSF.gff")
> msaList <- list()
> for (i in 1:length(mafFiles)) {
+   smallMsa <- read.msa(mafFiles[i], seqnames=seqnames)
+   smallMsa <- strip.gaps.msa(smallMsa)
+   smallMsa$offset <- 0
+   feat <- nrsfSites[which(nrsfSites$feature == nrsfNames[i]),]
+   if (feat$strand == "-")
+     smallMsa <- reverse.complement.msa(smallMsa)
+   msaList[[nrsfNames[i]]] <- smallMsa
+ }
> aggMsa <- concat.msa(msaList)
> motifLen <- unique(sapply(msaList, ncol))
> if (length(motifLen) != 1L) warning("all motifs should have same length!\n")
```

Notice the minor detail here of setting the "offset" of each alignment to 0. This is needed because alignments read in MAF format contain information about the coordinate offset of the alignment fragment relative to the chromosome in question. This information needs to be erased before the alignments are concatenated together.

Now we will create a feature set that labels each column in the concatenated alignment with the corresponding motif position (1–21). We can then easily estimate a model for each motif position with a single call to phyloFit. When we pass our feature set to phyloFit, it estimates a model for each feature type by default. Since we do not have very much data, we will not attempt to estimate a full model, but simply a scale factor and a set of equilibrium frequencies for each position.

```
> feats <- feat(seqname="hg18", src=as.character(sapply(nrsfNames, rep, motifLen)),
+               feature=rep(sprintf("site.%i", 1:motifLen), length(mafFiles)),
+               start=1:ncol(aggMsa), end=1:ncol(aggMsa))
> mods <- phyloFit(aggMsa, init.mod=neutralMod, no.opt="ratematrix",
+                  features=feats,
+                  scale.only=TRUE, ninf.sites=10)
```

Do not worry about the warning message that is generated by the call to phyloFit. This warning occurs simply because there are no sites in category "0" which is implicitly included in any feature set to represent the absence of any label (it is the "background" category).

Let us now summarize the estimated models in two ways. First, we will compute a (log) likelihood ratio comparing the fit of the estimated phylogenetic model to the fit of the neutral model at each position in the motif. This will give us a sense for how much phylogenetic information we have per motif position.

```
> nullLike <- numeric()
> lr  <- numeric()
> for (i in 1:motifLen) {
+   nullLike[i] <- likelihood.msa(concat.msa(lapply(msaList, `[`, cols=i)), neutralMod)
+   lr[i] <- mods[[i]]$likelihood - nullLike[i]
+ }
> barplot(lr, names.arg=1:motifLen, ylab="likelihood ratio")
```

The position-specific likelihood ratios can be seen in Figure 1. Let us compare these likelihood ratios with the known NRSF motif. We will use the "seqLogo" package to create a "logo" plot of the motif based on a motif model from TRANSFAC (included in the "examples.zip" package). The plot is shown in Figure 2.

```
> require("seqLogo")
> m <- read.table("NRSF/NRSF.mtf")
> pwm <- makePWM(t(m))
> seqLogo(pwm, xfontsize=10)
```

Notice the general agreement between the log odds scores and the information content across the positions of the motif. This suggests that sites with strong base preferences also tend to be conserved across mammalian species.

Second, let us describe the estimated motif based on on the equilibrium frequencies in the estimated phylogenetic models (called "background" frequencies in PHAST). We can also plot these using "seqLogo" (Figure 3):

```
> m <- matrix(0, nrow=21, ncol=4)
> for (i in 1:21)
+   m[i,] <- mods[[i]]$backgd
> pwm <- makePWM(t(m))
> seqLogo(pwm, xfontsize=10)
```

Again, we see a reasonable agreement between the estimates from these binding sites and the motif from TRANSFAC, despite the use of a relatively small data set.

Now we are ready to construct our phylo-HMM for binding site prediction. We will construct a 22-state phylo-HMM, with 21 states for the 21 positions in the motif, and a 22nd state representing "background" sequences. We will use a simple parameterization for the state-transition matrix, with a probability λ of entering a binding site from the background state, a probability of $1 - \lambda$ of remaining in the background state, a probability 1 of entering each motif position from the previous position, and a probability 1 of returning to the background state from the last motif position. Here we will simply set λ to a small value. Similarly, we will set the equilibrium frequency for the background state to a large value and the equilibrium frequencies for the motif states to small values. In a real application, we might wish to estimate these frequencies from training data.

```
> mods[["neutral"]] <- neutralMod
> get.trans.mat <- function(lambda, state.names, motifLen) {
+   transMat <- matrix(0, nrow=length(state.names), ncol=length(state.names),
+                      dimnames=list(state.names, state.names))
+   transMat["neutral", "site.1"] <- lambda
+   transMat["neutral", "neutral"] <- 1 - lambda
+   for (i in 1:(motifLen-1))
+     transMat[sprintf("site.%i", i), sprintf("site.%i", i+1)] <- 1
+   transMat[sprintf("site.%i", motifLen), "neutral"] <- 1.0
+   transMat
+ }
> lambda <- 0.0001
> transMat <- get.trans.mat(lambda, names(mods), motifLen)
> nrsfHmm <- hmm(transMat)
```

As a very simple demonstration of how this model can be used for prediction, let us generate simulated data containing binding sites, and then attempt to predict the locations of those binding sites using the HMM. We will use the same model to generate the data and make the predictions. This will obviously make the prediction problem somewhat easy, but it will at least give us a sense for the best-case performance of the method.

We will begin by simulating an alignment with 100,000 columns.

```
> simLength <- 100000
> simData <- simulate.msa(mods, simLength, hmm=nrsfHmm, get.features=TRUE)
```

Now let us predict binding site locations using the Viterbi algorithm. The "score.hmm" function will also collect marginal posterior probabilities for all states at all alignment positions, as we will show below.

```
> hmmScores <- score.hmm(msa=simData$msa, mod=mods, hmm=nrsfHmm, viterbi=TRUE,
+                        states=sprintf("site.%i", 1:motifLen))

Running Viterbi algorithm...
Scoring predictions...
Computing posterior probabilities...
Done.
```

We can now compare the predicted elements with the "true" elements generated by the simulator.

```
> predicted <- hmmScores$in.states
> correct <- simData$feats[substr(simData$feats$feature, 1, 4)=="site",]
> numSitePredicted <- coverage.feat(predicted)
> numSiteCorrect <- coverage.feat(correct)
> numSiteOverlap <- coverage.feat(predicted, correct)
> cat(numSitePredicted, numSiteCorrect, numSiteOverlap, "\n")

300 315 280
```

Observe that a large fraction of the predictions are correct (the exact number will depend on your randomly generated data set).

Next, we can use the "coverage" function to examine the sensitivity, specificity, and positive predictive value of the predictions. We can also plot the predictions alongside the "true" binding sites in a track. We will plot the position-by-position posterior probabilities together with the true and predicted binding sites (Figure 4).

```
> wholeRegion <- feat("hg18", src=".", feature="all",start=1, end=simLength)
> sensitivity <- coverage.feat(correct, predicted)/coverage.feat(correct)
> specificity <- coverage.feat(correct, predicted, wholeRegion, not=c(TRUE, TRUE, FALSE))/
+   coverage.feat(correct, wholeRegion, not=c(TRUE, FALSE))
> ppv <- coverage.feat(correct, predicted) /
+   coverage.feat(predicted)
> cat(specificity, sensitivity, ppv, "\n")

0.9997994 0.8888889 0.9333333

> tracks <- list(as.track.feat(correct, "actual binding sites"),
+                as.track.feat(predicted, "predicted binding sites", col="red"),
+                as.track.wig(coord=hmmScores$post.prob.wig$coord,
+                             score=hmmScores$post.prob.wig$post.prob,
+                             name="hmm Posterior probilities",
+                             smooth=FALSE, col="red", ylim=c(0,1)))
> plot.track(tracks)
```

While the prediction performance in this example is excellent, predicting binding sites in real data is considerably more difficult. In a real application, we would probably need a more complex model, for example, allowing for both conserved and nonconserved background states. Nevertheless, this example should provide a general idea for how a phylo-HMM can be created and applied using RPHAST.

Finally, we should be sure to clean up the "NRSF" directory.

```
> unlink("NRSF", recursive=TRUE)
```

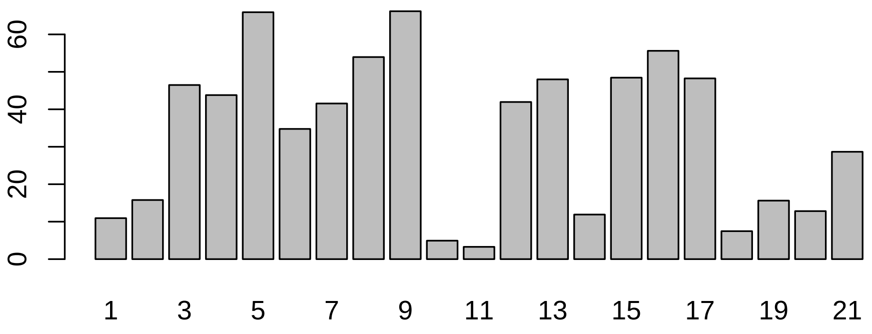

Figure 1: Likelihood ratio for conservation for alignment columns in a given motif position.

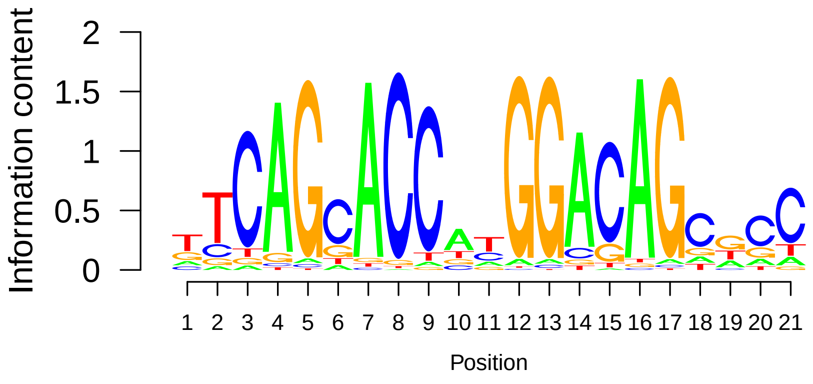

Figure 2: NRSF motif from TRANSFAC, plotted by the seqLogo package.

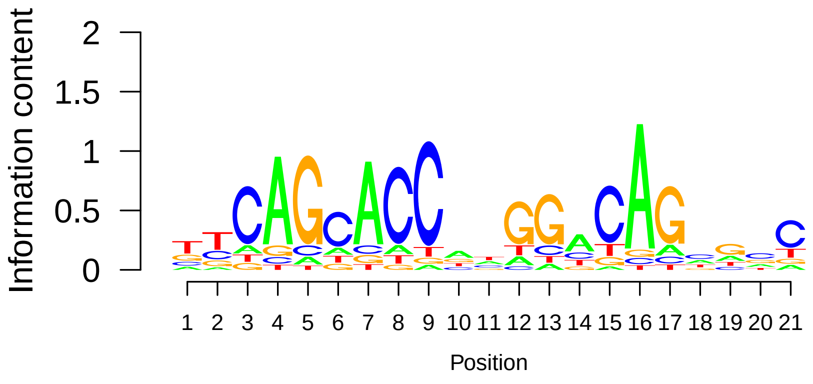

Figure 3: NRSF motif determined using background frequencies estimated by phyloFit, and plotted by the seqLogo package.

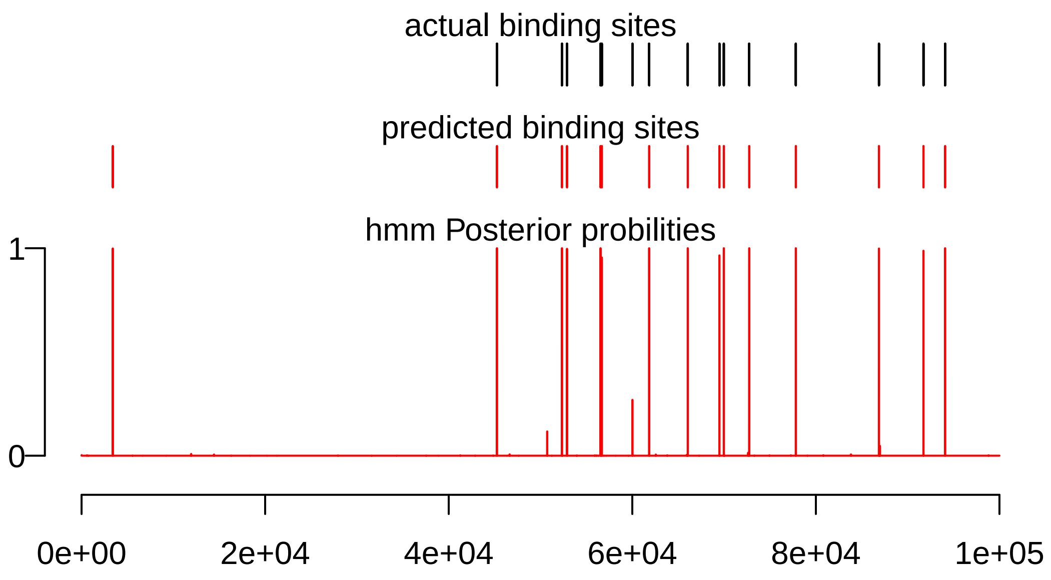

Figure 4: Browser-like plot showing locations of simulated binding sites, predicted binding sites, and posterior probabilities computed by the score.hmm function.