Conservation Analysis

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

December 22, 2010

Here we will show an example of conservation analysis in a non-model organism, *S. lycopersicon* (tomato). An alignment of tomato, potato, eggplant, pepper, and petunia is available through the table browser at Cornell's UCSC genome browser mirror (`http://genome-mirror.bscb.cornell.edu`). The necessary files are also included in the rphast package's example data set.

We begin by initializing RPHAST, loading the necessary data files, and defining the tree topology for the species of interest.

```
> require("rphast")
> # extract alignment and annotation files from RPHAST package
> exampleArchive <- system.file("extdata", "examples.zip", package="rphast")
> unzip(exampleArchive, c("sol1.maf", "sol1.gp"))
> # read alignment
> align <- read.msa("sol1.maf")
> # read gene annotations from a UCSC "genepred" file
> feats <- read.feat("sol1.gp")
> # define tree using Newick string
> tomatoTree <- "((((tomato, potato), eggplant), pepper), petunia);"
```

Next, we have to address some naming issues. The alignment makes use of Genome Browser-style sequence names (e.g., "sol1"), but it will be more convenient for us to use common names. The sequence names in the alignment must match those in the tree. In addition, the sequence name used for the gene annotations must match the name of the corresponding sequence in the alignment file (now "tomato").

```
> names(align)

[1] "sol1"    "solTub1" "capSp1"  "petSp1"  "solMel1"

> align$names <- c("tomato", "potato", "pepper", "petunia", "eggplant")
> names(align)

[1] "tomato"   "potato"   "pepper"   "petunia"  "eggplant"

> unique(feats$seqname)

[1] chr2
Levels: chr2

> feats$seqname <- "tomato"
```

Now let us augment the gene annotations. When reading a genepred file, RPHAST will create exon and CDS features only (with exons including both UTRs and coding regions, following the convention of GFF files). We would like to add features for introns. These particular gene annotations actually do not contain UTRs (they are derived from computational gene predictions, not mRNA/EST data), so the CDS and exon features are identical, and we can simply remove the exon features. Finally, we will add features explicitly defining intergenic regions.

```
> feats <- add.introns.feat(feats)
> feats <- feats[feats$feature != "exon",]
> table(feats$feature)

   CDS   exon intron
   204      0    153

> # make a feature that represents the entire chromosome.  We will ignore
> # several thousand bases at the beginning of the reference genome for
> # which no alignments are available by setting the "start" of the feature
> # equal to the beginning of the aligned region
> wholeChrom <- feat(seq="tomato", src=".", feature="all",
+                    start=align$offset,
+                    end=align$offset+ncol.msa(align, "tomato"))
> # annotate intergenic regions
> intergenicFeats <- inverse.feat(feats, region.bounds=wholeChrom)
> intergenicFeats$feature <- "intergenic"
> feats <- rbind.feat(feats, intergenicFeats)
```

Next we will estimate a neutral model from fourfold degenerate (4D) sites in coding regions, using phyloFit with the predefined tree topology and the general reversible (REV) substitution model.

```
> align4d <- get4d.msa(align, feats)
> neutralMod <- phyloFit(align4d, tree=tomatoTree, subst.mod="REV")
```

PhastCons can now be used to predict conserved elements, based on the estimated neutral model. For now, we will use the same values for the "expected.length" and "target.coverage" parameters that were used in the original analysis of these alignments (Wang et al., Genetics 180:391-408, 2008). Below we will consider an alterative way of setting these parameters.

```
> pc <- phastCons(align, neutralMod, expected.length=6,
+                 target.coverage=0.125, viterbi=TRUE)

Creating 'conserved' and 'nonconserved' states in HMM...
Running Viterbi algorithm...
Scoring predictions...
Computing posterior probabilities...
Done.

> names(pc)

[1] "most.conserved" "post.prob.wig"  "likelihood"

> consElements <- pc$most.conserved
> # this shows how many bases are predicted to be conserved
> coverage.feat(consElements)

[1] 6272

> # this shows the fraction of bases covered by conserved elements
> coverage.feat(consElements)/coverage.feat(wholeChrom)

[1] 0.0677512

> # the posterior probabilities for every base are here:
> names(pc$post.prob.wig)
```

```
[1] "coord"     "post.prob"

> dim(pc$post.prob.wig)

[1] 80646     2

> # and the overall likelihood is here:
> pc$likelihood

[1] -225284.0
```

For comparison, we will produce an alternative set of conservation scores using phyloP.

```
> pp <- phyloP(neutralMod, align, method="LRT", mode="CONACC")
> # the returned object is a data frame giving statistics for every base
> # in the alignment
> names(pp)

[1] "coord"    "scale"    "lnlratio" "pval"     "score"

> dim(pp)

[1] 80646     5
```

Let us now plot the gene annotations, conserved elements, and conservation scores for a genomic segment of interest. We will make use of functions in RPHAST that allow "tracks" to be defined and then plotted in a browser-like display.

```
> codingFeats <- feats[feats$feature=="CDS",]
> geneTrack <- as.track.feat(codingFeats, "genes", is.gene=TRUE)
> consElTrack <- as.track.feat(consElements, "phastCons most conserved", col="red")
> phastConsScoreTrack <- as.track.wig(wig=pc$post.prob.wig,
+                                     name="phastCons post prob", col="red", ylim=c(0, 1))
> phyloPTrack <- as.track.wig(coord=pp$coord, score=pp$score, name="phyloP score",
+                             col="blue", smooth=TRUE, horiz.line=0)
> plot.track(list(geneTrack, consElTrack, phastConsScoreTrack, phyloPTrack),
+            xlim=c(60000, 68000), cex.labels=1.25, cex.axis=1.25, cex.lab=1.5)
```

The plot produced by the code above can be seen in Figure 1.

Observe that the conserved elements appear to follow the coding exons reasonably well, but they also contain some noncoding regions. The (smoothed) phyloP scores are fairly consistent with the phastCons scores, but show some differences. Interestingly, the phyloP scores dip below zero just upstream of the gene of interest, indicating a region of accelerated evolution.

Now let us examine the predicted conserved elements in more detail. We will start by plotting their length distributions. We will plot distributions for all elements, and for the subsets of elements that primarily fall in coding or noncoding regions.

```
> ce <- pc$most.conserved
> plot(density.feat(ce), ylim=c(0, 0.018),
+      main="Element Length by Type", xlab="Length",
+      mgp=c(1.5,0.5,0),mar=c(2,2,2,2))
> # obtain elements that overlap codingFeats by at least 50 percent
> codingConsEle <- overlap.feat(ce, codingFeats, min.percent=0.5)
> # obtain elements that overlap by less than 50 percent
> noncodingConsEle <- overlap.feat(ce, codingFeats, min.percent=0.5,
```

```
+                                overlapping=FALSE)
> lines(density.feat(codingConsEle), col="red")
> lines(density.feat(noncodingConsEle), col="blue")
> legend(c("All", "Coding", "Noncoding"), x="topright", inset=0.05,
+        lty=1, col=c("black", "red", "blue"))
```

The above code produces the plot shown in Figure 2. We see that the coding elements are clearly longer on average, as expected. The coding elements appear to dominate the length distribution for all elements.

Now let us examine the relationship between conserved elements and annotations of different types. First, we will plot the fold-enrichment of annotation types within conserved elements, as compared with the genomic segment as a whole. Second, we will plot the composition of conserved elements by annotation type, and compare it with the "background" composition for the entire region.

```
> par(mfrow=c(2, 2), cex.main=1.5, cex.lab=1.5, cex.axis=1.5, mar=c(5,5,4,2))
> # look at fold-enrichment of each annotation type by conserved element
> enrich <- enrichment.feat(ce, feats, wholeChrom)
> col <- rainbow(nrow(enrich))
> barplot(enrich$enrichment, col=col,
+         main="Enrichment of\nConserved Elements",
+         ylab="Fold Enrichment")
> plot.new()
> legend(x="center", legend=enrich$type, fill=col, cex=1.5)
> # look at the composition of the conserved elements
> comp <- composition.feat(ce, feats)
> pie(comp$composition, col=rainbow(nrow(comp)), radius=1.0,
+     main="Composition of\nConserved Elements", labels=NA)
> # compare with background composition
> comp <- composition.feat(wholeChrom, feats)
> pie(comp$composition, col=rainbow(nrow(comp)), radius=1.0,
+     main="Background\nComposition", labels=NA)
```

The plots are shown in Figure 3. We see that coding regions are enriched more than fourfold in conserved elements, but introns are slightly depleted and intergenic regions are quite strongly depleted. The pie charts show clearly that CDSs are strongly overrepresented in conserved elements relative to background.

Next, we will run phastCons again, but this time instead of fixing the the transition probabilities between the conserved and nonconserved states by specifying the "expected.length" and "target.coverage" parameters, we we will allow it to estimate these probabilities by maximum likelihood. It does this using an expectation maximization (EM) algorithm.

```
> pcEM <- phastCons(align, neutralMod, viterbi=TRUE, estimate.transitions=TRUE)

Creating 'conserved' and 'nonconserved' states in HMM...
Finding MLE for (mu, nu)...
(mu = 0.012268. nu = 0.000646)
Running Viterbi algorithm...
Scoring predictions...
Computing posterior probabilities...
Done.

> names(pcEM)

[1] "transition.rates" "most.conserved"   "post.prob.wig"    "likelihood"

> pcEM$transition.rates
```

```
[1] 0.0122684734 0.0006457091
```

```
> pcEM$likelihood
```

```
[1] -224214.8
```

Note that the results now includes a "transition.rates" field, in addition to the ones that were present in the first run. Note also that the likelihood is somewhat higher that it was above, as it should be because the rates were estimated by maximum likelhood.

Let us now compare the coverage with and without estimation of the transition probabilities. We can use the "coverage.feat" function to examine the intersection properties of the two sets of elements. We can also plot them along the genomic region for visual comparison.

```
> coverage.feat(pcEM$most.conserved)
```

```
[1] 21576
```

```
> coverage.feat(pcEM$most.conserved, pc$most.conserved)
```

```
[1] 6155
```

```
> coverage.feat(pcEM$most.conserved, pc$most.conserved, or=TRUE)
```

```
[1] 21693
```

```
> coverage.feat(pcEM$most.conserved, pc$most.conserved,
+               not=c(FALSE, TRUE))
```

```
[1] 15421
```

```
> coverage.feat(pcEM$most.conserved, pc$most.conserved,
+               not=c(TRUE, FALSE))
```

```
[1] 117
```

```
> plot.track(list(as.track.feat(pc$most.conserved, name="No estimation"),
+                 as.track.feat(pcEM$most.conserved, name="With estimation")))
```

Observe that the coverage of the new elements is considerably higher. Most bases covered by the original elements are also covered by the new ones, but the reverse is not true—the new elements are largely a superset of the old ones. The browser plot of both sets of elements can be seen in Figure 4.

By comparing the length distributions, we can see that the new elements tend to be considerably longer, on average.

```
> plot(density.feat(pc$most.conserved),
+      main="Distribution of Element Lengths", xlab="Length", xlim=c(0, 1000))
> lines(density.feat(pcEM$most.conserved), col="red")
> legend(x="topright", inset=0.05, c("without estimation", "with estimation"),
+        lty=1, col=c("black", "red"))
```

The length distributions appear in Figure 5. What has happened here? It turns out that maximum likelihood estimation tends to produce small transition probabilities in the HMM, which leads to long conserved elements, containing many nonconserved as well as conserved bases. While this approach leads to higher likelihoods, it is generally not as useful biologically. In general, we recommend using the "expected.length" and "target.coverage" parameters instead, and tuning them in an appropriate way. For more discussion of this issue, see the phastCons "HOWTO" (http://compgen.bscb.cornell.edu/phast/phastCons-HOWTO.html) and the Supplementary Material of the phastCons paper (Siepel et al., Genome Res. 15:1034-1050, 2005).

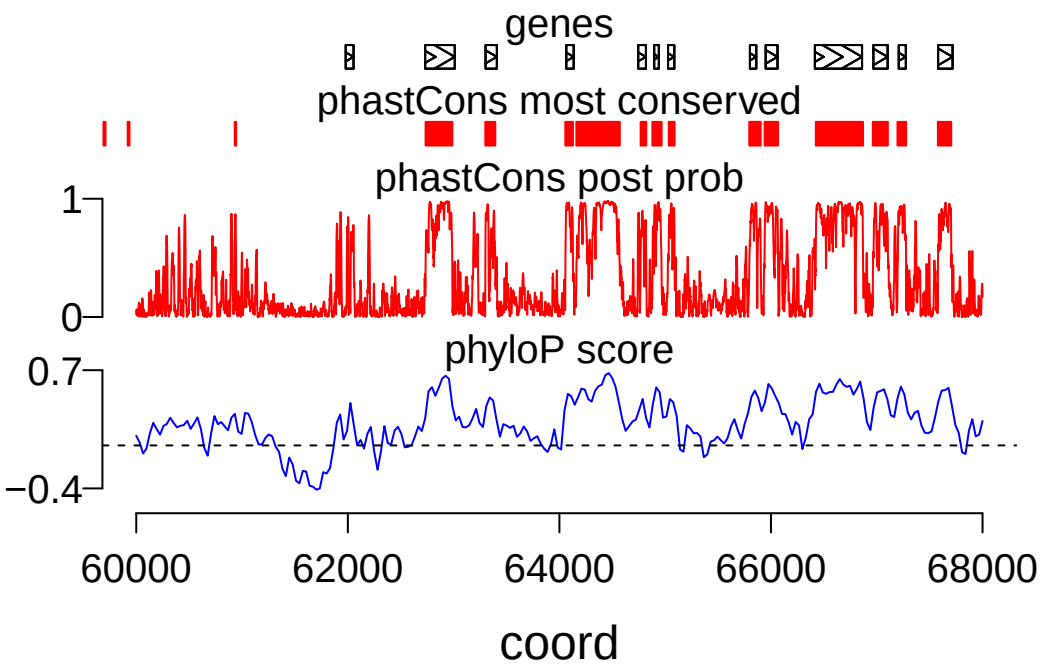

Figure 1: Browser-like display created by the plot.track function.

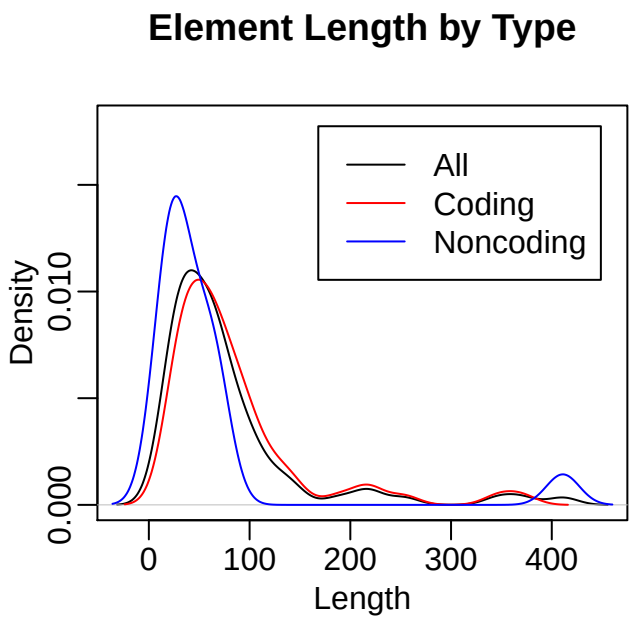

Figure 2: Distribution of the length of conserved elements.

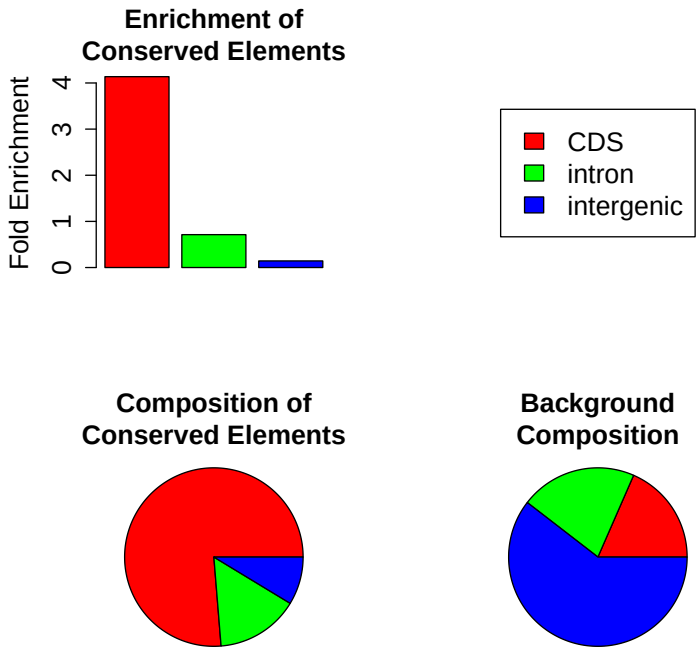

Figure 3: Composition of conserved elements.

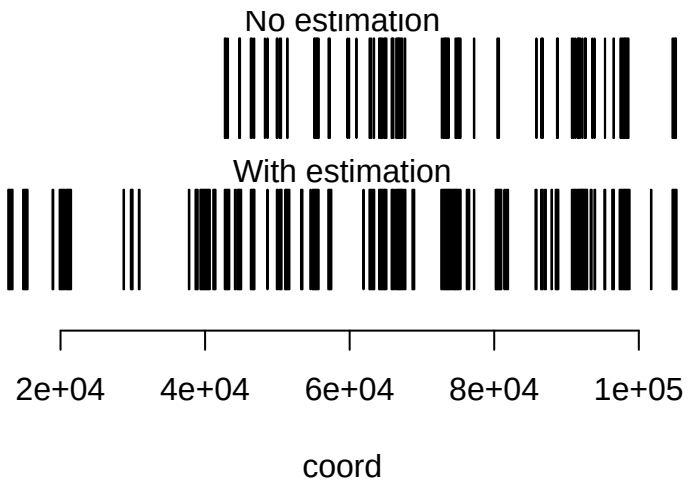

Figure 4: Comparison of conserved elements with and without estimation of transition probabilities.

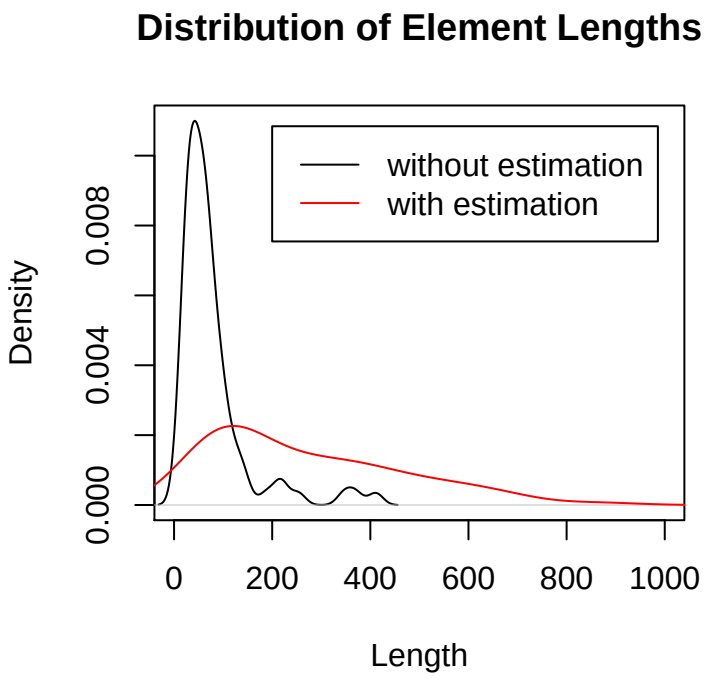

Figure 5: Distribution of element lengths with and without estimation of transition probabilities.