Detection of Accelerated Regions

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

December 22, 2010

Note: The source code for this example is not included in the rphast package (because the run time is too long for routine CRAN checks), but is available at the RPHAST webpage: `http://compgen.bscb.cornell.edu/rphast/vignette2.Rnw`.

This example will go through the basic steps of detecting accelerated evolution in a subtree. In this case, we will look for accelerated regions in the mouse-rat subtree, focusing on alignments from human chromosome 22.

The ideal way to perform this analysis would be to first identify conserved elements using the same procedure as in example 1, but in a subset of the alignment which excludes mouse and rat. Then one could examine the full alignments for signs of acceleration in mouse-rat in the identified elements. However, to keep the example simple we will make use of precomputed conserved elements from the UCSC genome browser.

The multiple alignments and conserved elements for this example can be obtained using the UCSC Table Browser for the hg18 assembly. We also make use of a neutral phylogenetic model from UCSC. For convenience, we have included the conserved elements and neutral model with the RPHAST package. We have also provided a direct link to the alignment file (see below).

We begin by initializing RPHAST and loading the neutral model and conserved elements. We have to change the feature names for the conserved elements to ensure that they match the name of the reference genome in the alignment ("hg18").

```
> require("rphast")
> exampleArchive <- system.file("extdata", "examples.zip",
+                               package="rphast")
> unzip(exampleArchive, c("placentalMammals.mod", "chr22-elements.bed"))
> neutralMod <- read.tm("placentalMammals.mod")
> elements <- read.feat("chr22-elements.bed")
> unique(elements$seqname)

[1] chr22
Levels: chr22

> elements$seqname <- "hg18"
```

Next, let's download the posted version of the alignment, which contains a subset of the full 44 species (to minimize download and compute time). The full MAF file can be downloaded through the UCSC Table Browser (under the hg18 assembly, Comparative Genomics Group, Conservation track, multiz44way table).

Since this is still quite a large alignment, we will use the pointer.only=TRUE option to read.msa. This option causes the alignment object to be stored on the C side, and handled only "by reference" (i.e., using a pointer) in R. It can be much more efficient than passing the entire object between R and C on every function call. However, some care must be taken when working with pointer objects in RPHAST, because, unlike ordinary R objects, their values may be changed as a result of a function call. When RPHAST functions change the value of arguments that are passed as pointers, this behavior is always documented in the help files.

```
> if (!file.exists("chr22.maf")) {
+   if (!file.exists("chr22.maf.zip"))
+     download.file("http://compgen.bscb.cornell.edu/rphast/chr22.maf.zip",
+                   "chr22.maf.zip", method="auto")
+   unzip("chr22.maf.zip", "chr22.maf")
+ }
> align <- strip.gaps.msa(read.msa("chr22.maf", pointer.only=TRUE))
> unlink("chr22.maf")
```

In this type of analysis, it is usually necessary to apply a strict set of filters to the alignments to avoid false positive "accelerated regions" that result from misalignments and other data quality problems. Typically, we would use synteny, duplication, and repeat filters, as well as filters to remove regions with too much missing data. Here we will implement a missing data filter.

The following code defines a set of features containing all positions that do not have missing data in mouse or rat. It then creates another set of features containing positions with data in a minimum of four species. We use the coverage.feat function to take the intersection of these two sets of features and the conserved elements. This produces a set of "informative elements" that pass the missing data filters.

```
> hasMouseRat <- informative.regions.msa(align, 2, c("mm9", "rn4"))
> hasAtLeastFour <- informative.regions.msa(align, 4)
> informativeElements <- coverage.feat(elements, hasMouseRat,
+                                      hasAtLeastFour, get.feats=TRUE)
> coverage.feat(informativeElements)/coverage.feat(elements)
```

```
[1] 0.78801
```

So 79% of the conserved elements have passed our informative elements filter.

We will also regularize the lengths of the conserved elements, to simplify our likelihood ratio tests. (Otherwise, we will have to consider the length dependency of the likelihood ratios when evaluating significance.) We will simply split the filtered conserved elements into fragments of size 50, and discard the leftover fragments:

```
> splitLength <- 50
> splitElements <- split.feat(informativeElements, f=splitLength,
+                             drop=TRUE)
```

Now we will run phyloP on these split elements. PhyloP will calculate a likelihood ratio for every feature, along with parameter estimates and approximate p-values (based on an assumed chi-square null distribution):

```
> obsPhyloP <- phyloP(neutralMod, msa=align, mode="ACC",
+                     features=splitElements, subtree="mm9-rn4")
```

For these short elements with a fairly small species set, we would like to compute empirical p-values, instead of relying on the assumption of a chi-square null distribution (which holds only asymptotically). We will use non-parametric simulations to calculate these empirical p-values. At the end of the vignette we will also show an alternative, parametric method for computing empirical p-values.

We begin by extracting the sites from the original alignment that correspond to the identified elements, using the "extract.feature.msa" function. Because this function will change a pointer object (as documented in the help file), we work with a copy of it. We then generate 5000 synthetic alignments by sampling (with replacement) from this set of sites. In real scenarios we would perform many more simulations (at least 100,000), but we choose 5000 here to make the vingette run quickly. It is also much more efficient to generate one long alignment, and interpret it as a concatenation of shorter alignments. We then run phyloP on this alignment to obtain our null distribution of log likelihood ratios. We take care to apply phyloP in exactly the same way as for the real data.

```
> elementAlign <- extract.feature.msa(copy.msa(align), informativeElements,
+                                     pointer.only=TRUE)
> nrep <- 5000
> simMsa <- sample.msa(elementAlign, nrep*splitLength, replace=TRUE)
> # produce features allowing long alignment to be interpreted as
> # concatenation of shorter alignments
> startIdx <- seq(from=1, by=splitLength, length.out=nrep)
> features <- feat(seqname=names.msa(simMsa)[1], src="sim", feat=".",
+                  start=startIdx,
+                  end=startIdx+splitLength-1)
> nonParaPhyloP <- phyloP(neutralMod, msa=simMsa, mode="ACC",
+                         features=features, subtree="mm9-rn4")
```

Let us compare the distributions likelihood ratios for the real and simulated (null) data sets using a Q-Q plot. We will also create density plots for both distributions, on the same scale.

```
> layout(matrix(c(1,2), nrow=2), heights=c(0.7, 0.3))
> par(mar=c(4.5, 4, 4, 2), mgp=c(2.5,1,0), cex.axis=1.5, cex.lab=1.5)
> qqplot(nonParaPhyloP$lnlratio,obsPhyloP$lnlratio,
+ xlim=c(0,15),ylim=c(0,15), xlab="Simulated likelihood ratio",
+       ylab="Observed likelihood ratio")
> abline(0, 1, lty=2)
> par(mar=c(4,4,1,2))
> plot(density(obsPhyloP$lnlratio,adjust=3), lty=1,xlim=c(0,15),
+      xlab="Likelihood Ratio",
+      ylab="Density",main="", col="red")
> lines(density(nonParaPhyloP$lnlratio,adjust=3), lty=1,
+ col="black",xlim=c(0,15))
```

The plots can be seen in Figure 1. Notice that the two distributions appear to be quite similar from the density plots. However, an excess of high likelihood ratios for the observed elements is clearly evident in the Q-Q plot.

Let us now calculate empirical p-values for the real data, and adjust them for multiple comparisons using p.adjust. We will use the Benjamini-Hochberg method to compute approximate false discovery rates (FDRs) for each element.

```
> empirical.pval <- function(x, dist) {
+   sum(x <= dist)/length(dist)
+ }
> nonParaPval <- sapply(obsPhyloP$lnlratio, empirical.pval,
+                       nonParaPhyloP$lnlratio)
> nonParaFDR <- p.adjust(nonParaPval, method="BH")
```

We will extract those elements having FDR < 0.05 and treat them as candidate "RAR"s (Rodent Accelerated Regions). We will write these elements to a file, which could be used, say, to display them as a custom track in the UCSC Genome Browser, or to perform an analysis using Galaxy. Notice that we have to change the sequence names to allow them to be recognized by the UCSC Genome Browser.

```
> nonParaSigFeats <- splitElements[nonParaFDR < 0.05,]
> nrow(nonParaSigFeats)

[1] 44

> nonParaSigFeats$feature <- "RAR"
> nonParaSigFeats$score <- obsPhyloP$lnlratio[nonParaFDR < 0.05]
```

```
> nonParaSigFeats$seqname <- "hg18.chr22"
> write.feat(nonParaSigFeats, "RAR.gff")
```

As a sanity check, let us estimate branch lengths for the RARs and compare them with estimates for the larger set of conserved elements. We will use the "ape" package to plot the estimated trees side-by-side (Figure 2). We can see that the RARs do show a pronounced evolutionary speedup in the rodents.

```
> consEleModel <- phyloFit(elementAlign, init.mod=neutralMod, no.opt=c("backgd", "rematrix"))
> rarAlign <- extract.feature.msa(copy.msa(align), nonParaSigFeats)
> rarModel <- phyloFit(rarAlign, init.mod=neutralMod, no.opt=c("backgd", "rematrix"))
> require("ape")
> par(mfrow=c(1,2), mar=c(5,2,4,2))
> maxX <- depth.tree(rarModel$tree, "mm9")
> plot.phylo(read.tree(text=consEleModel$tree), x.lim=c(0, 0.5),
+            main="Conserved Elements")
> plot.phylo(read.tree(text=rarModel$tree), x.lim=c(0, 0.5),
+            main="RAR")
```

We made use of phyloP in this example because it happens to be optimized for tests of clade-specific acceleration or conservation. However, we might want to perform a similar likelihood ratio test based on other models available in PHAST, but not already supported by predefined functions for likelihood ratio tests. Below we show how the same test could be implemented by making two calls to phyloFit, corresponding to the null and alternative models. The R function below could be used in place of the call to phyloP.

```
> acc.test <- function(msa, mod, subtreeNode) {
+   nullMod <- phyloFit(msa, init.mod=mod, scale.only=TRUE,
+                       no.opt=c("backgd", "rematrix"),
+                       quiet=TRUE)
+   altMod <- phyloFit(msa, init.mod=mod, scale.only=TRUE, scale.subtree=subtreeNode,
+                      no.opt=c("backgd", "rematrix"),
+                      quiet=TRUE)
+   tNullSub <- branchlength.tree(subtree(nullMod$tree, subtreeNode, super.tree=FALSE))
+   tAltSub <- branchlength.tree(subtree(altMod$tree, subtreeNode, super.tree=FALSE))
+   if (tNullSub >= tAltSub) return(0)
+   altMod$likelihood - nullMod$likelihood
+ }
> # example function call
> llr <- acc.test(sub.msa(align, start.col=splitElements[1,"start"],
+                         end.col=splitElements[1,"end"],
+                         refseq="hg18"),
+                 neutralMod, "mm9-rn4")
```

An alternative approach for characterizing the null distribution of log likelihood ratios would be to generate data from a parametric model. Below we demonstrate this approach. For our null model, we use a scaled version of the neutral model, with a scale factor estimated from the conserved elements. In estimating this scale factor, we exclude the rodents, because these are the species we wish to test for acceleration.

```
> seqs <- names(align)
> keepSeqs <- seqs[seqs != "mm9" & seqs != "rn4"]
> prunedMod <- neutralMod
> prunedMod$tree <- prune.tree(neutralMod$tree, keepSeqs, all.but=TRUE)
> subAlign <- sub.msa(elementAlign, seqs=keepSeqs)
> superTreeMod <- phyloFit(subAlign, init.mod=neutralMod,
```

```
+                          scale.only=TRUE, no.opt=c("backgd", "ratematrix"))
> superTreeScale <- branchlength.tree(superTreeMod$tree) /
+   branchlength.tree(prunedMod$tree)
> superTreeScale
```

```
[1] 0.2327303
```

```
> scaledNeutralModel <- neutralMod
> scaledNeutralModel$tree <- rescale.tree(neutralMod$tree,
+                                         scale=superTreeScale)
> # also remove species from model which are not in alignment
> scaledNeutralModel$tree <- prune.tree(scaledNeutralModel$tree,
+                                       seqs=seqs, all.but=TRUE)
```

Based on this model, we can now generate synthetic alignments using the simulate.msa function. We can then run phyloP on these alignments in the same way we did for the alignments simulated by the nonparametric approach.

```
> simMsa <- simulate.msa(scaledNeutralModel, nrep*splitLength)
> paraPhyloP <- phyloP(neutralMod, msa=simMsa, mode="ACC",
+                      features=features, subtree="mm9-rn4")
```

Let us see how different the RARs look, if we compute p-values based on this null distribution instead.

```
> paraPval <- sapply(obsPhyloP$lnlratio, empirical.pval,
+                    paraPhyloP$lnlratio)
> paraFDR <- p.adjust(paraPval, method="BH")
> paraSigIdx <- which(paraFDR < 0.05)
> length(paraSigIdx)
```

```
[1] 80
```

The results of this comparison will be somewhat different each time the code is executed, because the empirical p-values depend on random sampling, but when we ran it we found that the nonparametric method identified more elements. In general, this method may be somewhat less conservative, perhaps because the parametric model is overly simple, by failing to allow for rate variation among sites, indels, missing data, and other features of the real alignments. But the number of samples here is quite small (only 5000), so stochastic effects may cause one run to be quite different from another.

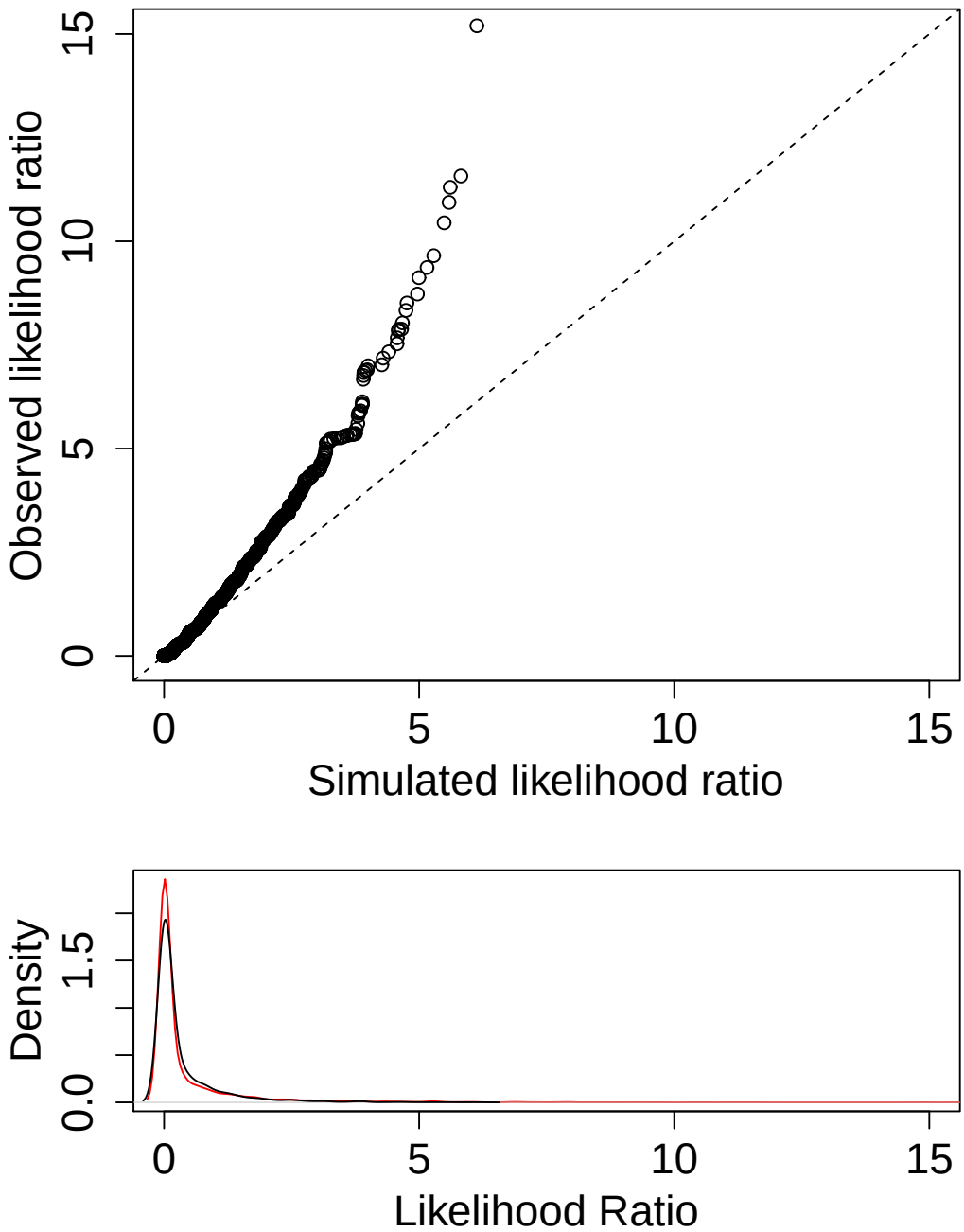

Figure 1: On top, Q-Q plot for observed and simulated likelihood ratios. On bottom, distributions of observed (red) and simulated (black) values.

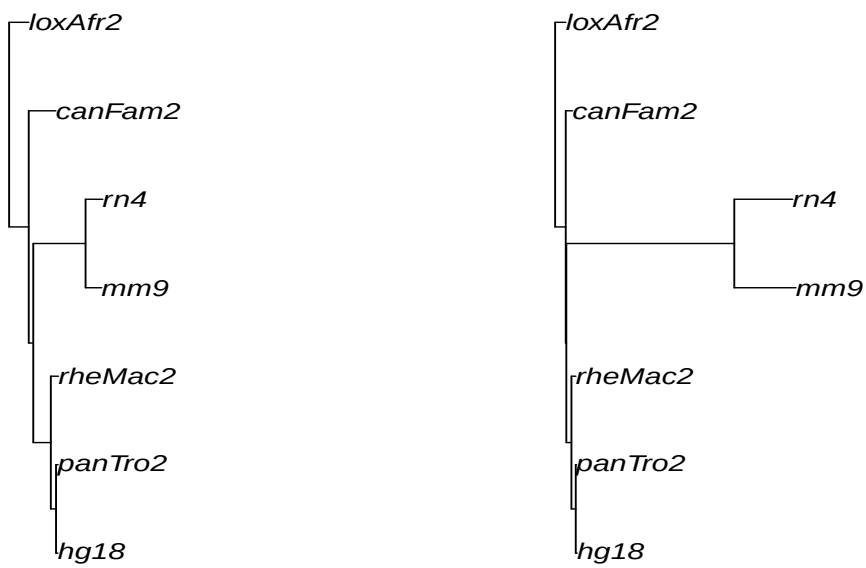

Figure 2: Trees estimated from conserved elements (left) and predicted rodent accelerated regions (right), plotted by the ape package.