IMPORTANT: For the PhastCons regions determined by the viterbi algorithm, there may be a few missing because where the maf files were split could be a single region that went uncalled. This is unlikely to affect much.

We recomputed PhyloP scores after masking various species for a variety of reasons.

To do this, we first installed everything with the instructions in installation.sh.

We then downloaded the hal file and maf file as in download.sh.

Next, we need to compute a .mod file for each split of interest. Our pipeline was heavily based on the code at this github repository from this paper: <https://github.com/michaeldong1/ZOONOMIA> 10.1126/science.abn2937 In this case, we wanted to compute one for the following subtrees. They are listed in the order Targets (subtree): Subtrees to check synteny against (used later).

Ape (PrimatesAnc015): PrimatesAnc025, 026

Old world monkey + Ape (PrimatesAnc007): PrimatesAnc015, 024, 040, 041

New world monkey + Old world monkey + Ape (PrimatesAnc003): PrimatesAnc015, 024, 040, 041, 016, 017

All primates (PrimatesAnc000): PrimatesAnc015, 014, 008, 004, 002

All mammals (fullTreeAnc239): fullTreeAnc5, 13, 115, 237, Primates000

**Note: Keeping with the original Zoonomia publication, we actually found repeats on the fullTreeAnc238 subtree then lifted over to the fullTreeAnc239 subtree.**

We used halStats to figure this out in conjunction with plugging the tree produced into <https://itol.embl.de/>.

We then used hal2fasta to get fasta files for each of the desired subtrees and move them into new directories, one for each fasta file.

We then went into those directories and called repeats using RepeatMasker. We used split\_fasta.py to split the fasta into ~2,000,000 line chunks, then ran RepeatMasker on each chunk using make\_sh.py to make the .sh files and driver.sh to run them. This took ~3 days to run.

Next we filtered out problematic repeats with filter\_repeats.py, checked for synteny against the branches above, selected ~100,000 random bases in repeats, and fit a model with PHAST PhyloFit. This generates a .mod file. We checked that the results were similar to the .mod files produced for the 241-way alignment in the paper from the github. We then lifted over to the human genome, selected 100 kb of repeat bases on chrY and chrX, and fit sex chromsome-specific models. These models can be found in “Mod\_files”. More details about the exact commands can be found in MakeModFiles\_<subtree>.sh. The related scripts are in Scripts/Making\_Mod\_Files

With the .mod files in hand, we were now ready to compute PhyloP/PhastCons scores. To do this, we split the .hal file into maf files per chromosome (referenced to hg38). We then computed on each individual chromosome. I will only include the example for chromosome 1 here because it was identical for all other chromosomes. This involved removing duplicates with mafFilter from mafTools, masking the fasta using phast maf\_parse, and then splitting into 5,000,000 base chunks with maf\_parse. This creates a lot of very large files. Similar to the mod files, we masked all non-primates, all mammals that were not monkeys or apes, and all mammals that were not old world monkeys or apes (as well as a no masking run). We then masked human, chimpanzee + bonobo, human + chimpanzee + bonobo, and human + chimpanzee + bonobo + gorilla + orangutan in order to test for positive selection/accelerated evolution in those lineages. This created a total of 20 maf files (including the one without any masking). We used make\_to\_mask.py on the chr21 file we initially made to do make one for each chromosome.

From there, we computed PhyloP scores, PhastCons scores/regions, and the number of species in the alignment at each base. We did this by creating a template file that does this, phylop\_phastcons\_447.sh, and then using make\_scripts.py to create one for each chrom/masking combination. E.g. for chr1 we would run

python make\_scripts.py chr1

This would create all the .sh files needed, a .sh file called move\_scripts.sh, and another file called driver.sh. We can then run ./move\_scripts.sh to move the scripts to the correct directories, and then run ./driver.sh to submit all the jobs. This relies on fix\_wig.py (which sets the chromosome name properly) and get\_species\_support\_447.py which computes the species support for each base (ie number of species in the alignment with unmasked bases).

Phast can often run into segmentation faults or other errors can occur or masking can fail for reasons unknown. To guard against this, we ran three scripts. The first, check\_error.py was run on each chromosome to check for error messages in the slurm.out files. The second, check\_wc.sh, printed the number of lines in each file to make sure there were the correct, consistent number (ie scores for all bases were computed successfully). Finally, check\_specsup.py checked to make sure that the maximum species support was what it should be given what species were masked in each file. For example, MskHum should be 447 - 1 = 446 species, MskNonMonkApe\_MskHumChpBonGorOrg should be 447 - 259 - 7 = 181 species, etc. (7 in the last one because of the two subspecies of orangutan/gorilla). This makes sure that the masking worked properly. All these scripts are in Scripts/Computing\_PhyloP\_PhastCons

After everything finished running, we catted together the output files from different chromosomes, gzipped them, and then backed them up to Computed\_Scores.