This is a detailed explanation of the pipeline (developed initially by Rachel M. Agoglia (RMA) and cleaned up/modified/rewritten by Alexander L. Starr (ALS)). ALS wrote this document and loosely adapted it from a document produced by RMA.

**Note 1:** While I tried to recreate Rachel’s pipeline exactly, there are very minor differences. The correlation between ASE values produced on the hybrids is 0.996 due to ~100 SNPs not being included or being altered in my version.

**Note 2:** Use the \_UCSC version after splitting (see useful commands at the end of this document) for mafs downloaded from UCSC.

**Note 3:** For information on which MAF files I used, see the file in Scripts\_For\_Downloading.

**Note 4:** To make file names sensible with other species, you will have to go into the scripts to change instances of “chimp”, “chp”, “huch”, and the file prefix from “Compara…” (if using ensembl mafs in cat.sh, INDELtxt.sh, and SNPbed.sh) to appropriate things. The files that need modifying are make\_indel\_bed.py, rmdup\_indels\_nhp.py, INDELtxt.sh/INDELtxt\_UCSC.sh, make\_bed.py, and SNPbed.sh/SNPbed\_UCSC.sh. Obviously you will have to change the contents of humanChroms.txt and chimpChroms.txt if you want to switch to refseq naming conventions. **However, with my rewrite swapping to refseq names is no longer necessary. This will make many downstream analyses much more convenient.**

**Note 5:** For more divergent species (pretty much any non-ape and gibbons), there will be >26 files formed from the split commands so you will need to copy and paste the for loops that have \*a\* in them and then change to \*b\*, \*c\*, etc. (in some cases out to \*e\* for very divergent species). Adjust the cat commands that rely on \*a\* appropriately as well.

**Note 6:** Mouse lemurs have 66 chromosomes (yikes) so I didn’t do any scaffold related filtering. You will just have to use them all, just toss the small ones and hope, or carefully figure out which ones are actually relevant.

**Note 7:** The splitByChrom.py script in its current form works specifically for ape vs. human comparisons that use UCSC chromosome naming conventions (chr1, chr2A, …, chrX, chrY). It can be modified by altering the list valid\_chrs appropriately. Chromosomes lacking chr are also supported.

**General Pipeline:**

1. The first step is to download .maf files (there will be several per chromosome) from either this ensembl link (they will need to be gunzipped and untarred with tar -xf <file\_name>: [Index of /ensemblgenomes/pub/metazoa/current/maf/ensembl-compara/pairwise\_alignments/ (ebi.ac.uk)](http://ftp.ebi.ac.uk/ensemblgenomes/pub/metazoa/current/maf/ensembl-compara/pairwise_alignments/) or from UCSC [UCSC Genome Browser Downloads](https://hgdownload.soe.ucsc.edu/downloads.html) (you can also find the liftover chain files here).

2. Next, run SNPbed.sh. This file is dependent on the files/scripts present in the folder with it. It uses callSNPsFromMAF.py to extract matched coordinates, a cat command to merge by chromosome, an awk line of code to extract actual SNPs, another cat to merge all the SNPs, a split call to split it into 5,000,000-line chunks for processing, make\_bed.py to reformat for filtering and swap bases to their complement when stranding switches, another cat call to merge again, and finally switchChrom.py to change the chromosome names to refseq. It may be necessary to use liftover if we want to use a different genome after this is done. **Note that the human allele is always the reference (ie human allele|chimp allele) in the SNP file (not the case for WASP files necessarily). This holds for the bam splitting scripts as well.** There is also an optional liftover step near the end. **Don’t do this.**

3. Once the bed file of SNPs is generated, it is ready for the filtering step. This requires mapped parental bam files with duplicates removed (usually called rmdup.bam).

4. To generate a WASP ready file of indels, use INDELtxt.sh. This works by calling indels from the maf file (directly), pooling all indels together, splitting into human and chimpanzee indels and switching to bed format/swapping to the complement when strands switch, **optionally lifting over**, switching back to the WASP format, sorting and filtering out duplicates, renaming the chromosomes to the desired naming convention, filtering out the duplicate indels that aren’t handled by uniq, and then using a bunch of awk lines to separate the files by chromosome. These are then ready to be merged with the SNPs for WASP that will be produced from SNP filtering. **Note: The strange way this is done ensures that the files match how Rachel did it to the greatest extent possible.**

**SNP Filtering:**

1. The first step in the pipeline for filtering is to run CountSNPASE.py (species specifically of course).  The first argument is the .bed file produced from running the above pipeline.  The second argument is a parental bam file containing mapped reads with duplicates removed, usually an rmdup.bam file.   The final argument is the desired outfile.  In all cases, it should start with snpCounts and contain a species-specific string (e.g. “H2”) that denotes the species the reads came from, **not** the species the reads were mapped to. **Need 256 GB of RAM.**

2. snpReview.py requires that all the files of interest (ie files aligned to the proper genome and with both NHP and human parental files) be present in the current directory and start with snpCounts.  It will also require changing of line 66 (the outfile line) to have a different name.  This file takes in all the snpCounts files **<see 1 of SNP Filtering>** and produces the file used by SNPreview.R for filtering.  It also has hardcoded into it things to denote species (“H2” for human and “C3” for chimp).  It will have to be heavily modified to deal with other species.

3. SNPreview.R produces the second input to the below step.  The only arguments to this function are GRCH38\_SNPreview.ALL.SPLIT\_SPECIES.txt and NHP\_SNPreview.ALL.SPLIT\_SPECIES.txt for human and NHP respectively **<see 2 of SNP Filtering>**.  The file only outputs a 2 column txt file with chromosome in the first column and the position of the SNP to keep in the second column.

4. sbatch\_snps\_pipeline.sbatch takes in the final output bed file of the generated SNP files from MAF files pipeline outlined in the commands document (ie ASE\_SNPs.bed for human) as its first argument **<see steps 1-3 of General Pipeline>**.  It then takes in a .txt file containing the final list of SNPs that should be included in the downstream analysis **<see 3 of this section>**.  The last argument is the file that will be used to store the passing SNPs and will be used downstream in splitting bam files and counting ASE. This uses SNPfilter.py. **The output of this script can be used in the count\_ase step of the snakemake pipeline for calling ASE or splitting BAM files in the splicing pipeline.**

**Creating Final WASP Files:**

1. The next step is to take the resulting filtered SNP file and run it through splitByChrom.py. This will output things on a per chromosome basis and format them for WASP, you will likely have to change some of the names in splitByChrom.py.

2. The final step is to merge the filtered SNPs with the indels previously generated. Depending on species, you pretty much have to write a bunch of cat commands. Examples are included in the Create\_Final\_WASP\_Files folder. **Congrats, you are now ready to correct mapping bias and do cool things with NHP/Human Hybrids!**

**Useful commands**: sed 's/...//' file <this removes the first three characters of a line (needed after liftover)>

sed 's/^/chr/' file.txt <this adds chr to the beginning of every line, necessary before liftover>

**Can use split <maf> -l 50,000 <species> to split the maf file downloaded from UCSC.**

**Extension:** Lately we have been using WGS data from the chimps/humans from which the cell lines were derived. This means that we retain ~75% of SNPs, far too many to load into 4 GB of RAM. Due to this, we must do More\_Goddamn\_Filtering.

1. There are 3 ways currently supported: retain all the SNPs in transcripts, retain all the SNPs in exons, and retain all the SNPs with >= min\_reads (by default equal to 2) in at least one SNPcounts file. For example, let’s say that you have 10 SNPcounts files and two SNPs, A and B. If A has one read supporting it in every file and B has two reads in one file and no reads in the other 9 files, A will be thrown out and B will be kept. I recommend using the latter method. To do this, take the SNPcounts files and put it in the same directory as get\_supported\_snps.py and change the hardcoded line 14 if needed. Usage is: get\_supported\_snps.py <species> <min\_reads> where species is the species that the original bamfiles and SNPfile were aligned to. get\_exon\_pos and get\_transcript\_pos work similarly. The output is a text file with the cooridnates of the SNPs with sufficient support, separated by chromosome, and sorted.

2. Next run splitByChromKeepAsBed <file.bed> <species> <suffix> with the first argument as the file to be split, the second argument as the alignment species, and the third as the suffix to name the files. This outputs a bed file separated by chromosome and sorted.

3. Finally, run filter\_by\_sorted <Suffix1> <Suffix2> <species> <t> <min\_reads>. Suffix1 is the suffix used in step 1, Suffix2 is the suffix used in step 2, species is the alignment species t is one of [transcript, exon, reads] depending on what the choice in step 1 was, and min\_reads is the min\_reads used in step 1 or anything at all if t != reads. The output is a filtered bed file which can then be plugged into the pipeline at Creating Final WASP Files above.