Lab Notebook

Postdoctoral Research

Sonal Singhal

sonal.singhal1@gmail.com

Beginning 7 July 2014

Contents

Monday, 7 July 2014 1
1 Variant calling. 1
2 Data management. 1

Tuesday, 8 July 2014 2
1 Data management. 2
2 Variant calling. 2
3 Making masked genomes. 2
4 Phasing variants. 3

Wednesday, 9 July 2014 5
1 Making masked genomes. 5

Thursday, 10 July 2014 6
1 Phasing variants. 6

Friday, 11 July 2014 8
1 Data management. 8
2 Phasing variants. 8

Weekend and Monday, 14 July 2014 10
1 Phasing variants. 10
2 Generating recombination map. 10

Tuesday, 15 July 2014 11
1 Phasing variants. 11
2 Generating recombination map. 11

Wednesday, 16 July 2014 12
1 Phasing variants. 12
2 Generating recombination map. 12

Thursday, 17 July 2014 14
1 Generating recombination map. 14
2 Phasing variants. 14
3 Data management. 15

Friday, 18 July 2014 16

1 Phasing variants. 16
2 Generating recombination map. 16

Weekend and Monday, 21 July 2014 **17**
1 Phasing variants. 17
2 Generating recombination map. 17

Tuesday, 22 July 2014 **18**
1 Phasing variants. 18
2 Generating recombination map. 18

Wednesday and Thursday, 23 and 24 July 2014 **19**

Friday, 25 July 2014 **21**

Monday, 28 July 2014 **22**

Tuesday, 29 July 2014 **24**

Wednesday, Thursday, Friday, 30 July - 1 August 2014 **25**

Monday, 4 August 2014 **26**

Tuesday, 5 August 2014 – Friday, 15 August 2014 **28**

Monday, 18 August 2014 **31**

Tuesday, 19 August 2014 **32**

Wednesday, 20 August 2014 **33**

Thursday, 21 August 2014 **34**

Friday, 22 August 2014 **35**

Monday, 25 August 2014 **36**

Tuesday, 26 August 2014 **37**

Wednesday, 27 August 2014 **38**

Thursday, 28 August 2014 **39**

Friday, 29 August 2014 **40**

Monday, 7 July 2014

1 Variant calling.

After talking with Molly, I looked at the weird peaky behavior of VQSR filter scores for long-tailed and double barred finch. All the weird peaks could be attributed to variants that were fixed or nearly fixed.

Also, I wrote code `count_triallelic_sites.py` to look at multi-allelic sites in long-tailed finch. It turns out that a significant portion of these sites are actually bi-allelic because the reference allele (or, the zebrafinch allele) is not at all represented.

2 Data management.

Issues with the `/KG/` drive continued. Started to migrate data off and explored ways to make one centralized repository for data. See emails with John Zekos for information on what happened and how to avoid it.

Tuesday, 8 July 2014

1 Data management.

All data but a bare minimum of final VCF files was moved off the Columbia server. BAM files for doublebarred finch and longtailed finch were moved off the `/KG/`drive and onto my more secure home drive.

2 Variant calling.

Looking at the multiallelic sites in longtailed finch found the following:

- 3.52% of LTF variable sites have 2 or more alternate alleles reported.
- 1.34% of LTF variable sites have 2 or more alternate alleles reported, but only 2 of the alleles were genotyped individuals – i.e., no individuals had the reference allele.
- 2.18% of LTF variable sites are sites that are truly multiallelic.

3 Making masked genomes.

Each run of the script `make_masked_genomes.py` produces two files, the masked genome and a summary file that tells the user how many of each site type was produced. This information is best summarized in the README, a portion of which is duplicated here.

```
The masked genome is represented as a FASTA file with each site given as a numeric, mutua

The sites are coded as following:
0: coverage within acceptable bounds; no variation
1: coverage within acceptable bounds; variant that is HQ and no evidence of Mendelian dis
2: coverage within acceptable bounds; variant that is HQ and has evidence of Mendelian di
3: coverage within acceptable bounds; variant that is low quality
4: unacceptable coverage; no variation
5: unacceptable coverage; variant that is HQ and no evidence of Mendelian distortion
6: unacceptable coverage; variant that is HQ and has evidence of Mendelian distortion
7: unacceptable coverage; variant that is low quality
```

```
Summing sites falling in categories 0 to 3 gives an estimate of total callable sites. Sum
```

I had made the doublebarred finch masked genome earlier last week. These were the files used.

```
vcf_file = '/mnt/lustre/home/sonal.singhal1/DBF/after_vqsr/DB.allchrs.vqsr.snps.indels.vc
cov_summary = '/mnt/lustre/home/sonal.singhal1/DBF/masked_genome/doublebarred_depth_summa
cov_data = '/mnt/lustre/home/sonal.singhal1/DBF/masked_genome/doublebarred_avg_depth.txt
mendel_file = \"\"
```

These are the files used to make the zebrafinch masked genome.

```
vcf_file = '/mnt/gluster/home/emleffler/genotype_callsets/zebrafinch/zf_unrels/unified_ge
cov_summary = '/mnt/lustre/home/sonal.singhal1/ZF/masked_genome/zebrafinch_depth_summary
cov_data = '/mnt/lustre/home/sonal.singhal1/ZF/masked_genome/zebrafinch_avg_depth.txt'
mendel_file = '/mnt/gluster/home/emleffler/genotype_callsets/zebrafinch/zf_family/unified
```

Note that in zebrafinch, coverage was based on unrelated individuals only, because related individuals were sequenced to higher depth. Also, in the same vein, only variants from the unrelated were considered. Is this going to be a problem?
These are the files used to make the longtailed masked genome.

```
vcf_file = '/mnt/gluster/home/emleffler/genotype_callsets/longtailedfinch/after_vqsr/gatk
cov_data = '/mnt/lustre/home/sonal.singhal1/LTF/masked_genome/longtailed_avg_depth.txt'
cov_summary = '/mnt/lustre/home/sonal.singhal1/LTF/masked_genome/longtailed_depth_summary
mendel_file = ''
```

4 Phasing variants.

To phase these chromosomes, we are going to use ShapeIt, first using their feature that lets you determine phase-informative reads (PIR) – these are mate-pairs that span two heterozygous sites. They tell you phase. Cool idea. However, to do this, we need VCFs that include SNPs and indels and that are by chromosome, and in this case, are also filtered by callable sites.

To do this:

1. Frustratingly, some VCFs are ordered by chromosome number (Chr1, Chr1A, Chr1B, etc) whereas others are ordered älphabetically(̈Chr10, Chr11, etc). This makes GATK and a number of programs unhappy, so the first thing I did was take a genome ordered alphabetically and create sequence dict and index files, using Picard and samtools respectively.

2. Then, it turns out that we didn't have a filtered VCF file for LTF that combined across both SNPs and indels and across the chromosomes. So, I wrote a little script `filtered_variants.py` that I used to pick out variable passing sites from the full genomic, all quality VCF from Ellen.

3. Then, I wrote a script that borrows from `make_masked_genome.py` and creates a VCF without inappropriate coverage (higher than $2\times$, lower than $0.5\times$ average genomic coverage) and splits it across chromosomes.

Wednesday, 9 July 2014

1 Making masked genomes.

It looks like all my masking of genomes has worked and that it is finally completed. Hurray! I put the README in each of the `masked_genome` folders and sent out the info to the group. Also, I copied the LTF directory to Columbia for Alva. Note that adding categories 0 through 3 gives the effective sequence length, which will be crucial for all other analyses.

To determine the final call sets, I took the (until that point) most final call set and removed sites with very low or very high coverage using the script `make_vcf_filtered_for_coverage_by_chr`
The VCFs used were the following:

- LTF: `/mnt/lustre/home/sonal.singhal1/LTF/after_vqsr/gatk.ug.ltf.allchrs.allvar.filter`
  - Note: I had to create this VCF, because for whatever reason, it did not exist already.
  - I created it using `filtered_variants.py` on `/mnt/gluster/home/emleffler/genotype_callse`
- ZF: `/mnt/gluster/home/emleffler/genotype_callsets/zebrafinch/zf_unrels/unified_genotyp`
  `/gatk.ug.unrelzf.allchrs.snps.indels.vqsr2.filtered.nomendel.recode.vcf.gz`

Thursday, 10 July 2014

1 Phasing variants.

I started my phasing experiments. There are four main ways that I could phase.

1. by using PIRs in ShapeIt and then LDhelmet
2. by using family information in ShapeIt and then LDhelmet
3. by using PIRs and family information in ShapeIt and then LDhelmet (is this even possible??)
4. by using LDhat straight away

I am going to pursue approaches 1 and 4 first, because I am still trying to figure out how to implement approach 3 and approach 2 seems like it would be less informative. Also, SNP calling for approach 2 was weird, and I am still getting my head around that.

In order to do approach 1, I got messed up by the extractPIRs program, but after some trial and error, I found:

- the pre-compiled version of extractPIRs works great; compiling on servers is hard because their version of g++ etc is so outdated
- you cannot have indels in the VCF file
- you cannot have non-biallelic SNPs in the VCF file
- VCFs have to be separated by chromosome
- although some of our multiallelic SNPs are really biallelic, I decided to just drop them, because keeping them would require recoding the VCF, which seems inadvisable
- I filtered the VCF files by using the script `remove_multialleles_indels.py`
- I really wanted to have proper BAM files for PIR calling, because it uses mate pair info. Due to the data problems, I didn't have one for LTF sample G118. I replicated KS's and EL's commands on LTF and moved forward with that.
- It turns out that ZF bam files and SNP calls are given two separate IDs, the intersection of which is in my local dir `/Users/singhal/zebrafinch/samples/zf_ids.txt`. Why would you do this?

Thursday, 10 July 2014

ShapeIT needs the BAM files to be indexed, so I did that, but it looks like G294 is truncated. So, copied that over again and indexed it, too.

I also realized that the ShapeIt program automatically defaults to N_e and ρ values for humans, which don't seem advisable for these birdies. So, I am going to do some work to approximate these values for birds.

Let's start with N_e. The easiest way to get a proxy estimate for N_e is to infer θ and to hope that the mutation rate estimate is reasonable. I looked through some papers, and the zebrafinch mutation rate has been reported as $2.21 \times 10^{-9} \frac{mutations}{site \cdot year}$ (Nam et al 2010; doi:10.1186/gb-2010-11-6-r68) and $2.95 \times 10^{-9} \frac{mutations}{site \cdot year}$ (Balakrishnan and Edwards; doi: 10.1534/genetics.108.094250). Zebrafinch have 3 - 4 generations a year, which means the per generation mutation rate is incredibly low – on the order of $7 \times 10^{-10} \frac{mutations}{bp \cdot generation}$. This paper (Ksepka et al. 2014; dx.doi.org/10.1098/rspb.2014.0677) suggests these estimates tend to predict much older divergence times than fossils, which makes me wonder if these mutation rates are downwardly biased. Plus, they are all based on the Zink calibration for mitochondrial rates, which I really wouldn't trust.

Anyways, I am going to go from θ estimates to N_e using Watterson's theta, so I need to calculate the number of segregating sites. (The VCF includes fixed sites and indels, which aren't appropriate for inclusion.) I wrote a script called `calculate_segregating_sites.py`.

Friday, 11 July 2014

1 Data management.

I lost most of the day to moving files around. I have 1 TB of space on the `/mnt/lustre/` drive and 1 TB of space on the `/mnt/glustre/` drive. That should be enough for everything, though it is definitely not ideal to have things spread out.

This is getting super annoying. This lab notebook is also starting to sound like a teenage diary.

2 Phasing variants.

I started my phasing experiments. There are four main ways that I could phase.

1. by using PIRs in ShapeIt and then LDhelmet
2. by using family information in ShapeIt and then LDhelmet
   a) I really cannot make sense of how this was done already. Should I just go ahead and use it, or redo it in a way that makes more sense?
3. by using PIRs and family information in ShapeIt and then LDhelmet
   a) An e-mail from Oliver Deleneau suggests that no, this is not possible. Bummer!
   b) I suppose a kludge-y way to do this would be to phase family and then use that as a reference
4. by using LDhat straight away

Honestly, I expect these results are going to be robust across analysis types. I should probably do some sort of power analysis to check and see what my PIR power is going to be – calculating the average distance between biallelic SNPs should tell me a lot. Also, should I look at some kind of four gamete test like they did in Mimulus? A lot of this gets output by ShapeIT, so I should just look at that. It is non-intuitive to understand the reporting results, but it is a decent proxy.

Back to calculating N_e. For zebrafinch, the total number of segregating sites is: 48726579. The total sequence length is: 859594837 bp. So, the segregating sites measure (S_n) is: $\frac{48726579}{859594837} = 0.0567$. Remember that $\theta = \frac{S_n}{a_n}$, where a_n is $\frac{1}{1+\frac{1}{2}+\frac{1}{3}...+\frac{1}{n-1}}$. Here, $n = 38$

because we have 19 diploid individuals. $a_{38} = 0.2317$, which means that $\theta = 0.2447$. $\theta = 4N_e\mu$, and let's take $\mu = 1 \times 10^{-9}\frac{mutations}{site \cdot gen}$, which means $N_e = 61 million$. That's absurd. If mutation rates are more on the order of humans, then we get a more reasonable $N_e = 6 million$. This actually concords quite well with the Balakrishnan and Edwards value of $N_e = 7 million$. Similarly, for longtailed finch, $S_n = \frac{27995773}{897015175} = 0.0312$ and $a_{40} = 0.229$, which means that $\theta = 0.136$. Assuming the same human-ish μ, $N_e = 3.4 million$.

Now for calculating ρ. I didn't know much about how ρ was calculated, or really what it meant. First, a centimorgan (c) is the distance between genes for which 1 out of 100 meiosis products are recombinant. Then, ρ is the expected number of crossover events per generation per base, and $\rho = 4N_e c$. The main tricky part of this equation is that c is the numerator of a fraction (the denominator is 100), so we need to take care of that and make it actual number before moving forward. In this case, the mean recombination rate in zebrafinches has been reported as $1.3\frac{CM}{Mb}$ (Backstrom et al 2009; 0.1101/gr.101410.109). Converting to bp and turning it back into a fraction, $c = 1.3 \times 10^{-8} meioticproductsperbp$. For zebrafinch, $\rho = 4 \cdot 6e6 \cdot 1.3e-8 = 0.312$, and for longtailedfinch, $\rho = 4 \cdot 3.4e6 \cdot 1.2e-8 = 0.1768$.

These numbers might all be balderdash, but they are certainly quite different from the defaults, so I'll go with them for now.

So went ahead and got started, and got the dreaded underflow in sequencing error. I am trying four things:

1. rerunning with more RAM (20g) – still failed
2. getting rid of window size – still failed
3. getting rid of window size and theta / rho
4. just getting rid of theta / rho

I am redoing the same file (chr23) that worked before, so something is probably off.

This redos showed that for all that work looking at theta and rho, that is what was bugging the program! Will run with the incredibly wrong default values. This seems like a bad idea. Well, I am going to go forth with the bad idea, but I e-mailed Deleneau first. Another user had seen the same error, so I posted an update to that message.

IN GENERAL I AM NOT HANDLING THE Z CHROMOSOME WELL. WILL NEED TO RECONSIDER THIS.

Weekend and Monday, 14 July 2014

1 Phasing variants.

This weekend, started running ShapeIt based on PIRs for LTF and finished up the runs that failed for ZF. Some runs for ZF failed because they had insuffiicient memory. It is worth noting that the cluster does not generate the standard ḿemory heapérror when there is no more memory; rather, it just sorta shuts down.

2 Generating recombination map.

In order to run LDhelmet, the file formats I have now need to be parsed in many ways. The first is that I need a mutation matrix, which shows the stationary distribution of mutation rates between different bases. To do this, I followed Chan et al. 2012 (10.1371/journal.pgen.1003090) and started to implement the method by writing the script `get_mutation_matrix.py`.

Tuesday, 15 July 2014

1 Phasing variants.

Some of the initial ZF ShapeIt runs failed – for a random subset of the smaller chromosomes. I reran those with larger window sizes, hoping it was an issue with window being too small.

2 Generating recombination map.

I finished the `get_mutation_matrix.py` script – it took some finagling of the reference genome so that it was in the same format as my masked genome files – the n̈ew referenceğenome is `reference/taeGut1_60.bamorder.fasta`. I am currently running it on the ZF genome. If that works, I will repeat with the LTF genome. It worked, so I did it with the LTF genome.

I spent a lot of time trying to figure out how to use the De Maio, Schlotterer, and Koisel (2013) method (PoMo) to get ancestral allele states. Along the way, I had to install Python and finagle with that and associated issues with libraries and such. It ended up wasting most of the day, because it turns out that PoMo does the ancestral allele reconstruction, but there is no way to get that information out of the program. Bummer.

I then went back to our original idea of using the outgroups to get the ancestral state, though an informal counting scheme, essentially. I started to implement this in `simple_ancestral_allele.`
There seems to be a lot of ancestral polymorphism, so it will be hard for this to be as exacting as I'd like. We'll see! Maybe it is fine given that everything is a prior, anyways.

Wednesday, 16 July 2014

1 Phasing variants.

Good progress on the phasing. Everything worked but for chromosome 20 in ZF. Pretty good! Need to figure out what happened there, but worth moving forward, at least.

2 Generating recombination map.

Another thought – maybe use the Darwin's finch to try phasing? It is about 10 mya diverged, which is pretty far, but is better than nothing. Will look at mapping efficiency, and go from there. Downloaded the reads from the SRA using the sra-toolkit – wow, this is much better than just FTPing the reads to the server. Will definitely want to use it in the future. Used the reads in Project PRJNA178982 in SRA binary format and then dumped into FASTQ format using fastq-dump.

In the meantime, I prepared the genome using Stampy. I am using Stampy rather than my favorite (bowtie2) because it can handle divergent reads well, which I imagine will be relevant here.

```
~/bin/stampy-1.0.23/stampy.py --species=zebrafinch --assembly=taeGut1 -G taeGut1.bamorder
~/bin/stampy-1.0.23/stampy.py -g ~/reference/taeGut1.bamorder -H ~/reference/taeGut1.bamo
/mnt/lustre/home/sonal.singhal1/bin/stampy-1.0.23/stampy.py -g ~/reference/taeGut1.bamor
```

Here are the long-awaited mutation matrices! I calculated these as followed by Chan et al 2013. These are both reported as A, C, G, T for both rows and columns, and the directionality is from row to column. These look a lot like the Drosophila matrices published in the Chan paper, though slightly different.

Here is the matrix for the zebrafinch.

$$\begin{pmatrix} 0.455 & 0.104 & 0.322 & 0.119 \\ 0.206 & 0.001 & 0.135 & 0.659 \\ 0.659 & 0.135 & 0 & 0.206 \\ 0.119 & 0.322 & 0.103 & 0.455 \end{pmatrix}$$

Here is the matrix for the longtailed finch.

$$\begin{pmatrix} 0.437 & 0.103 & 0.344 & 0.117 \\ 0.205 & 0 & 0.151 & 0.644 \\ 0.644 & 0.151 & 0 & 0.205 \\ 0.117 & 0.344 & 0.103 & 0.436 \end{pmatrix}$$

Wednesday, 16 July 2014

These two matrices look pretty similar, as I would think.

Thursday, 17 July 2014

1 Generating recombination map.

Aligning the ground finch genome sequences to the zebrafinch genome was going very slowly, and Stampy doesn't work in parallel unless you are running in BWA mode, so I created my own stupid parallel Stampy by splitting up the read file into 22 subfiles, and then running Stampy on 22 processes. It is still going pretty slowly, but at least this is a 22x speedup. I will then combine all the SAM files to get one master result, whenever it is finished.

I worked on getting the filtered for quality and coverage VCF files for longtailed finch into LDhat format. The main constraints of the program is that it can only handle variable, biallelic positions, so I had to get rid of any fixed or true polyallelic positions. Another fussy thing about LDhat – it cannot handle multiple variants at a site, so I had to dump any such variants. All of these tend to be when GATK calls a SNP and indel for the same position. I did this using the script `convert_vcf_to_ldhat.py`, which works on the files (all variants, not just biallelic) in the `/mnt/lustre/home/sonal.singhal1/LTF/after_vqsr/by_`

For LDhat, I need to get a likelihood lookup table, which is contingent on the θ for the species and the N (number of chromosomes) sampled for the species. To do that, I ran the `complete` program that is part of LDhat. The command I ran was:

```
~/bin/LDhat_v2.2/complete -n 40 -rhomax 100 -n_pts 101 -theta 0.136 -prefix LTF
```

The θ value I used was the same one I calculated using the number of segregating sites (Watterson's θ).

2 Phasing variants.

I am not sure why chromosome 20 for zebrafinch keeps failing, so I am going to try rerunning extractPIRs with greater stringency for quality, and see if that makes a difference. New command:

```
~/bin/extractPIRs.v1.r68.x86_64/extractPIRs --bam /mnt/lustre/home/sonal.singhal1/ZF/phas
```

Although it didn't influence the number of PIRs found or used, this run finished. Doesn't make sense, but I suppose it need not to.

Thursday, 17 July 2014

I also wanted to try running ShapeIt using the duoHMM option, which allows you to use any level of pedigree information to phase chromosomes. Ellen had used an older version of ShapeIt that only allowed one trio or one duo. So, to do that, I had to first properly merge the unrel_zf and rel_zf files, because for some reason, they were called independently of each other. I am not sure why. To do this, I did the following:

1. Took as the starting VCF for the un_rel zf to be `/mnt/gluster/home/emleffler/genotype_callsets/`
   Note that this includes all sites; I need to be able to distinguish between non-variable sites and sites with missing data, and this is the only way to do that.
2. Took as the starting VCF for the rel zf to be `/mnt/gluster/home/emleffler/genotype_callsets/ze`
   Note that this includes all sites; I need to be able to distinguish between non-variable sites and sites with missing data, and this is the only way to do that.
3. Took the un_rel zf VCF and separated it by chromosome and filtered it for coverage and non-mendel sites using `make_vcf_filtered_for_coverage_by_chr_no_mendel.py`.
4. Took the rel zf VCF and separated it by chromosome and filtered it for coverage and non-mendel sites using `make_vcf_filtered_for_coverage_by_chr_no_mendel.py`.
5. I then started writing a script that will take these two VCFs and merge them into a PED Plink output. I am not using VCF format because it would require me to recode the VCF to better handle indels and ̈fakep̈olyallelic sites, and I don't want to do that.
6. VCF to ShapeIt PED format
   a) family id (unique for every individual)
   b) individual id
   c) father
   d) mother
   e) sex
   f) phenotype??
   g) Genotypes
      - 1 - indel 1
      - 2 - indel 2
      - A,T,C,G, as expected
      - 0 = missing

3 Data management.

I set up my education GitHub account, which allows me to have private repos. I set up two repos, one for my scripts (`https://github.com/singhal/postdoc`) and the other for my lab notebook (`https://github.com/singhal/labnotebook`).

Friday, 18 July 2014

A bit of a slow day due to medical appointments.

1 Phasing variants.

I wrote the script that will take the two sets of VCFs for zebrafinch (related and unrelated) that contain all sites that pass the coverage filter. From these files, it will combine to only select out the sites that are polymorphic in either set of variants. I will then run these through CombineVariants in GATK.

2 Generating recombination map.

Okay, I totally messed up LDhat in two ways. First, I didn't account for the fact that it is best run as 4000 sites at a time, so I changed the `convert_vcf_to_ldhat.py` script to account for running 4000 sites with 200 bp overlap on either side, except for edge conditions. In the process of deleting the old files, though, I managed to delete the likelihood lookup table! Grr. So I am rerunning the likelihood lookup table, and once that is done, I will start the convert script if it works with a test file. This is the command I tried on my original files, and then realized these files are too big and would take way too long to run.

```
~/bin/LDhat_v2.2/rhomap -seq filtered.coverage.vqsr_chr22.sites -loc filtered.coverage.vc
```

One good thing about this: I found Adam Auton's 2012 Science paper (10.1126/science.1216872), and it has the best SOM – very clear and should give me some guidance on things that I should consider.

Weekend and Monday, 21 July 2014

1 Phasing variants.

Molly wants us to try phasing including the sites that have three alleles or more. I am not so sure about the logic of this, given that LDhelmet can only handle biallelics. Perhaps the thought is that it will help phasing, even if some of the sites don't make it to the next round? Anyways, the extractPIR routine of ShapeIT cannot handle VCFs that have polyallelic sites, so I had to recode any such sites as two biallelic sites. I wrote a script `change_vcf_polyalleles_bialleles.py`, which does that and also filters for indels ... which extractPIRs cannot handle for sure. I am now re-running the extractPIR routine on all chromosomes for ZF. I have kept the previous results in the `phasing/PIR_approach/old/` folders under each species folder. I cannot rerun them for LTF because the BAM files have moved. AGAIN.

Finding the intersection of sites between the unrel and related ZF individuals finally completed, so I am going to concatenate the files now. I concatenated them using the script `run_GATK_CombineVariants.py`. Note that anytime you run GATK, you need to make sure that they have been compressed with bgzip and indexed with tabix.

2 Generating recombination map.

As part of the ongoing attempt to get good ancestral allele site info, the alignment to the finch genome finally finished, and now I am working on converting all the SAM files into sorted BAM files... I will then combine across BAM files so that I have something that I can use with GATK for variant calling. So, I converted, sorted, and combined, and am now using samtools and bcftools to call the genome. The command I ran was
`~/bin/samtools-0.1.19/samtools mpileup -I -uf ~/reference/taeGut1_60.bamorder.fasta Geos`
and was borrowed liberally from the PSMC suggestion (https://github.com/lh3/psmc).

Tuesday, 22 July 2014

1 Phasing variants.

Ellen put all the LTF bams in the same place `/mnt/gluster/data/internal_restricted_supp/finches_2` so I started the next round of ShapeIt calls.

I then spent a lot of time trying to get ShapeIt for families to work. First, I finished the script `convert_vcf_to_ped.py`. It changed a lot since I first wrote it, because I wanted to account for the revised way that I intersected variants. It turned out that intersecting variants that way was a bit of a disaster, because although I output only sites that were polymorphic, I didn't account for sites that were polymorphic for a SNP but not for an indel. So, I had to make sure to only select the sites that were PASSing in one or the other VCF file, even though you would think this merged VCF would only have those sites. Then, it turns out that ShapeIt in the family mode (duoHMM) cannot handle triallelic sites, nor can it handle triallelic sites that are coded as two biallelic sites. (There went 2 hours.) So, I had to rewrite the script to only consider biallelic sties, and when there was a site with both a SNP and an indel, I kept the SNP and dropped the indel, because the next downstream step (LDhelmet) can only use SNPs. The basic test command I ran was `~/bin/shapeit_v2r790/shapeit --input-ped chr16.ped chr16.map --duohmm -W 5 --output-m` I will want to finesse it. Worth noting that the chromosome names have to be recoded a bit for ShapeIt. The recoding was the following: chr1: 1, chr2: 2, chr3: 3, chr4: 4, chr5: 5, chr6: 6, chr7: 7, chr8: 8, chr9: 9, chr10: 10, chr11: 11, chr12: 12, chr13: 13, chr14: 14, chr15: 15, chr16: 16, chr17: 17, chr18: 18, chr19: 19, chr20: 20, chr21: 21, chr22: 22, chr23: 33, chr24: 34, chr25: 35, chr26: 36, chr27: 37, chr28: 38, chr1A: 30, chr1B: 31, chr4A: 32, chrLG2: 39, chrLG5: 40, chrLGE22: 41, chrZ: X. Also, that sex is 1 for ZW and 2 for ZZ.

I tested the LDhat likelihood table and it worked. But like an idiot, I deleted it again. Am recreating it and will run it tomorrow. I am also convinced that I should be running the interval program of LDhat instead of rhomap, so will adjust that accordingly. The test files of 4000 SNPs each worked, though, so I am going ahead and creating those using `convert_vcf_to_ldhat.py`.

2 Generating recombination map.

There was a problem with how finch genome was recreated, in that I allowed indels. Uf. Am redoing it without indels.

Wednesday and Thursday, 23 and 24 July 2014

This was a bit of a hodge-podge kind of day. First, it turned out that my `convert_vcf_to_ped.py` had a computing leak, so it took me a lot of sleuthing to find it. It turns out that I had used a list where I really should have used a dictionary, and that slowed down the script to a snail's pace once it had parsed half the rows. That said, this script still requires tons and tons of memory, because it stores all the genotypes in memory. I cannot figure out an easier way to do this that handles all bialllelic sites properly (this prevents me from using Plink), so I am just running the script with ton of memory. But, because bigmem01 is down, these runs have been slow to go.

Then, I played around some with LDhat. It turned out that the way the manual suggested to name chromosome length was ambiguous, and I had made the wrong decision in how to parse it. So, I had to fix up my script (`convert_vcf_to_ldhat.py`) to do it properly and run it again. I played around with interval, the program that I will be using instead of rhomap, and it looks like the key parameter is this block penalty parameter, which defines how many blocks of recombination there are in the set of SNPs. Auton recommends using 5, but he also recommends testing multiple values. I am not sure how I will do that...

Then, I worked on the ancestral genomes simple method. In my test case, about 60% of sites were pretty easy to unambiguously assign to an ancestral condition – these were sites for which the other two finch species had no variation. That's pretty good. For the other cases, I started thinking about using the Geospiza genome. Once I started to look at it, though, I was very disappointed by the quality of the SNP calls in the Geospiza genome. Further investigation showed that a very small percentage of the Geospiza reads aligned to the zebrafinch genome (¡20%) even though the original paper reported alignment rates of 80%. So, I decided that Stampy is, in fact, as bad of a program as it seems. I went back to using Bowtie2 when some test cases showed that it was giving alignment rates nearer to 80%. I wrote the tiniest script to do this called `run_bowtie.py`. It did look like some the runs failed partially through. I am not sure if this was a memory issue or what, so I reran those. Otherwise, this script seems like a pretty decent way to define ancestral alleles, though the true measure will be inspecting the unfolded SFS (and maybe even folded SFS).

Also, some of the ShapeIt post-PIR runs failed, so I am trying to run them again, removing the window call. That worked for almost all the failed runs last time.

Wednesday and Thursday, 23 and 24 July 2014

I continued the work on the ancestral allele identification, and I took a look at the SFS generated from about 10K SNPs. Comparing it to the expected SFS (this work is on my local computer, `sfs.ipynb`), there are more singletons than expected (and fewer low-frequency variants) and there is a slight uptick for high-frequency derived alleles. Bummer. This is the mark of not well-polarized SNPs. Will take a second look at the code and see what I can do to avoid it.

I also started to get another outgroup sequence from Geospiza fortis, which is closely related to Geospiza magnirostris, so really won't add any additional information ... however, the benefit is that this sequencing was done to much higher coverage and with Illumina data (so less prone to indel related errors). Will be an additional confirmation. The project ID in NCBI is PRJNA156703 and the p̈aperïs http://gigadb.org/dataset/100040. Remember that the path here is prefetch to fastq-dump to gzip FASTQ to Bowtie to SAM to sorted BAM to call variants.

Friday, 25 July 2014

Today, I ended up sending a lot of time on `birdtree.org`, in an attempt to get a phylogeny for the species that I am studying. This is based on the Jetz paper from 2013 in Nature, and the website allows you to download a subsample of the trees generated through the MCMC chain, and you can then summarize across the trees.

Monday, 28 July 2014

I continued work on getting the Geospiza fortis reference sequence – I started the alignments over the weekend, and they are going slowly. All the sequence online will give over 100× coverage, which is more than I need, so I am probably going to limit the alignment at a certain point. Okay, about midway through these runs, I finally checked the SAM files and they were super weird. That's because the folks who uploaded these reads decided to concatenate head-to-tail the two paired ends of a read. Uf. I had to write a script (`splitreads.py`) to split each read up and to put it into two separate reads, and I will need to figure out the more appropriate Bowtie commands for these reads now.

Over the weekend, I also started the family runs for ZF ShapeIt, implemented in `run_shapeit_fam.py`. The basci command is: `shapeit --input-ped PED_FILE MAP_FILE --duohmm -W 3 --output-max HAPS`

So of all the chromosomes for the family-based phasing, all are working well ... except for chromosome 26. I just spent three hours trying to track down why it isn't working, and I have no clue. ShapeIT is saying that there is a SNP with completley missing data, but there is no such evidence. (It is important to note that this error is only thrown in duoHMM mode.) I thought the problem could be with my script, so I tried to create the MAP and PED files using vcftools and plink, but that recreated file threw the same error. I posted to the ShapeIt server to see what they thought. For now, I will just use Ellen's results, perhaps?

I kept on working on `simple_ancestral_alleles.py` – no big changes, though it remains that about 10% of sites are ambiguous. I hope that adding G. fortis will improve things, though we will see.

I started the LDhat runs for long-tailed finch. I decided to use interval instead of rhomap, based on reading some of Auton's other papers. I split up all the calls (8219 in total!) across 10 different runs, which I just started. Also, to make things more reasonable, I am running them for only 5e6 iterations. I also am using a block penalty of 5; knowing no better what to use, that's what Auton recommended in the manual.

I started to filter the DBF files for multiallelic sites and indels, using `remove_multialleles_indels.py`. There was a problem with DBF having SNPs and indels at the same site, but this will get rid of that problem because it will remove all indels. I think this is kind of cheating, but I have also noticed that almost no programs use indels at all for anything. I think indels are generally untrusted.

Monday, 28 July 2014

When I tried to use the filtered VCFs for ShapeIt haplotype calling, it failed because ShapeIt refuses to use just less than 10 individuals for phasing, unless you have a reference panel. Am going to try again ReadBackedPhasing from GATK, which I've used before and didn't find all that great, but is better than nothing. Got ReadBackedPhasing going, using this basic command `/home/shyamg/bin/java -Xmx4g -jar /mnt/lustre/home/sonal.sing`

Tuesday, 29 July 2014

The splitting of the FASTQ reads for G. fortis finished, so I started the Bowtie2 reads. It seems to be going well, and pretty quickly.

Some of the DBF readbacked phasing failed, for no apparent reason. Could this be connected to the instability of oss3? Started these again with slightly more memory.

Wednesday, Thursday, Friday, 30 July - 1 August 2014

Lost a lot of time due to the move and lab visitors.
Interesting conversation with Iain Matheison, who argued a few things:

1. You should always use the aligner designed for that variant caller (BWA w. GATK, Stampy w. Platypus)

2. It is okay that I couldn't use m̈ore accurateëstimates of ρ and N_e for ShapeIT – the program has been calibrated to run at those values, and other values make it fussy. (That's exactly what I saw.)

3. When calculating switch error rate, probably a lot of my errors are due to singletons or doubletons (and other rare variants) being out of phase. I might want to discount such sites, because phasing them will be very hard, unless they are covered by a phase-informative site. He isn't that surprised by my high switch error rates.

4. I might want to phase and make the recombination map only using non-rare variants. Whew.

5. Given my high switch error rate, it is worth considering running LDhelmet with both phasing attempts and just running LDhat on unphased data.

6. I should really pursue the combined approach for phasing.

Monday, 4 August 2014

So through a lot of trial-and-error, I independently come onto a way to calculate error rates in phasing that is widely known in the literature as s̈witch error rate.̇ This basically counts how many times you have to switch the phasing to get the true and fake haplotype to match up. The denominator is the number of sites that you have to phase, which is the number of heterozygous sites for that individual. In Amy's work, where they were phasing tons of genomes (1000×, the switch error rate tends to be quite low - less than 10%). In my samples, the switch error rate is really high, especially when there are few heterozygous sites. The average switch error rate is more like 30%. Some of this is likely due to rare variants, as Iain Mathesion pointed out. But, the point of all this is it makes me very wary about this phasing, particularly because we haven't accounted for uncertainity in phasing. I think that I will account for this by running LDhat and by trying to use haplotypes from both phasing methods ... and possibly by trying to combine the phasing methods. The script was implemented in `compare_haplotypes.py`.

I installed LDhelmet, which took a while because the Boost libraries are pretty fussy. The only way I could get the install to work was to edit my `.bashrc` profile to include `export LD_LIBRARY_PATH=/mnt/lustre/home/sonal.singhal1/bin/lib/`. This command has to be executed every time I run LDhelmet so that it can find the libraries it needs to run.

Throughout the weekend, I worked on sorting and merging the BAM files from G. fortis. Things were super fussy, because it turns out I couldn't get any of the multi-threaded options to work on the cluster, which I kinda wanted to use, because these files were so big. Finally, I just went and did it the normal way. Once I had the sorted and merged bam file (resulting file: `Geospiza_fortis.bam`, I went back to create the reference genome sequence. I used (essentially) the same command I used for the G. magnistrosis genome: `echo "~/bin/samtools-0.1.19/samtools mpileup -I -uf /mnt/gluster/home/sona`
It appears to be running impossibly slowly, but I suppose that is because it is such a big BAM (70g).

I moved over most of my files to `/mnt/gluster/home/sonal.singhal1/` because J. Zekos said the lustre file system was running out of space. Then I asked him to change my home directory to be that directory, but that didn't turn out to work so well. He had to change it back. I ended up loosing quite a bit of time in that. Most importantly, I had to stop the long-running LDhat runs and start them again.

And oh – I figured out why the chr26 for ZF family kept failing because of a missing data error. (There were no obvious data missing.) It was because Mendelian errors in `-duoHMM` mode lead to imputation at those sites, which could lead to a site with variation becoming a site with no variation. To deal with that, I removed all sites with Mendelian errors and reran. It worked!

Tuesday, 5 August 2014 – Friday, 15 August 2014

Very few updates here because I had to backtrack a lot over the last two weeks. These two weeks were essentially dealing with my concerns that the switch error rate inferred between family-informed phasing and read-backed phasing was exceptionally high – it could be as high as 30% to 40% across any given chromosome. This clearly would not do, as the basis for recombination estimates revolves around inferred breakpoints between haplotypes.
I met with Molly, and she was skeptical about everything, including:

1. Using human-specific values for ρ and N_e; she advised using the bird specific values (which are different from the ones I inferred). Although that still wouldn't work because of the overflow errors, I did increase the rho value to be 0.0008, or double the default. This makes it more in line with the values of c reported in the literature, assuming the default N_e in the program? But Molly said it should be $10\times$ greater? I am not sure why or how.

2. The percentage of heterozygous sites covered by PIRs. I don't quite get this concern, though I suppose it would be easy enough to test empirically.

3. The influence of singleton mutations in switch error rate – they don't make that big of a difference, it would turn out.

4. That PIR phasing would be better than family phasing, which was my instinct but not hers.

5. My implementation of the ancestral allele finding, because I only considered the Darwin's finches when the more internal branches were uninformative. For this concern, I think it is best to be aware of the inconstinency and code sites accordingly.

So, I went back and took a closer look at phase uncertainty and what might be causing it and how I could handle it. These explorations are covered in my iPython Notebook on my local drive called `phasing_error.ipynb`. Basically, ShapeIt allows uncertainty to be realized two ways – you can either sample sets of haplotypes from the haplotype graphs, or you can output the frequency of haplotypes at a given set of sites. Either way, you get a sense for how much general uncertainty there is, but very little appreciation for how any given site has been tricky to phase. I thought (mistakenly) that one could get phasing uncertainty at pairs of sites, and then knock out those sites that do not

have good phasing certainty. Yeah, that didn't work. First of, it turns out that even modest decreases in phasing uncertainty can result in appreciable rates of switch error; it doesn't take much for things to get wonky. Second of, phasing isn't really something you can do site by site (though this is confusing because that's how Phase used to report the results); rather, it is something that you do in context of other sites. Knowing how phasing uncertainty propagates along a haplotype seems impossible too predict from casting a look at more regional windows. Given all this, I decided the best bet is to go with the most likely haplotypes and then get an idea for the switch error rate along a chromosome. If it turns out that we see interesting patterns, we can either sample across all haplotypes at a given hotspot (for example) and see if the pattern holds, or we can confirm that switch error rate is low in that region.
The rest of the time I spent backtracking on the work that I had done, because I realized that I really disagreed with recoding the triallelic sites to be two biallelic sites, especially because it lead to a lot of överflow errors.. Also, it made an insignificant difference. So, starting with LTF, I did the following:

1. Started with the by chromosome files that had been filtered for coverage.
2. Recoded f̈ake polyallelesäs bialleles and removed any indels to create the `*recoded_biallelicSNPs` files.
3. Took these VCFs and used them for PIR extraction and assembly, using the higher rho value AND outputting the haplotype graphs (these capture the uncertainty around haplotype estimation).
4. Used the most likely haplotypes to create the sequence files (40 sequences for the population because each individual donates two haplotypes) using the script `make_seq_for_ldhelmet.py`. Importantly, I removed any singletons, because the phasing uncertainty profiling showed that removing singletons reduced switch error rates.
5. Found the assumed ancestral allele at every variable site using the script `simple_ancestral_alleles`
6. Ran the script that calculates phasing uncertainty for every set of 50 SNPs.

Starting with ZF, I did the following:

1. Started with the raw, unfiltered files for unrelated and related zebrafinch.
2. Filtered both for coverage and divided them up into chromosome.
3. Combined the unrelated and related individuals.
4. Filtered for variants.
5. Turned into PED / MAP files.
6. Used PED / MAP files with Plink to identify non-Mendel sites.

7. Removed non-Mendel sites from VCF files.
8. Turned into new PED / MAP files.
9. Ran with family-style `--duohmm`. Looked at switch error rate among samples from the haplotype graph. I only looked at the family because smart folks had suggested to me that those should phase perfectly. Not at all the case. Switch error rate was incredibly high – 5% to 40%! Decided that the switch error rate for the family was too high and I could not use them AT ALL as my reference haplotype set, as I had intended.
10. Took the non-mendel, filtered VCF files and recoded them so that f̈ake polyallelicsẅere biallelics and removed indels.
11. Took these and started the PIR process.

Monday, 18 August 2014

Turns out all the jobs I had submitted for the assemble mode of ShapeIt failed with the dreaded overflow error. I looked at the jobs for a while, and it come to seem that was because I included the family samples. So, I backtracked my analyses to only include the unrelated zebrafinches, which required me to create the `*filtered.recoded_biallelicSNPs*vcf` files from the coverage filtered files. After that, I restarted the PIR finding, as standard, and then restarted the assembling. Slowly but surely!

The other big hassle of today was trying to figure out why the `make_seq_for_ldhelmet.py` script was so slow. It essentially wasn't running for some chromosomes, which, it turns out is because I was having lists append to lists, which grows like $O(n)$. Lists are really funky with computational cost, so I need to be more careful how I use and employ them. I fixed it though.
Spent some of the day working on my graduate school project on the Eugongylus skinks.
I also identified the ancestral alleles for LTF using the following files.

```
vcf_in = '/mnt/gluster/home/sonal.singhal1/LTF/after_vqsr/by_chr/gatk.ug.ltf.%s.filtered
vcf_out1 = '/mnt/gluster/home/sonal.singhal1/ZF/after_vqsr/by_chr/gatk.ug.all_zf.%s.cover
vcf_out2 = '/mnt/gluster/home/sonal.singhal1/DBF/after_vqsr/by_chr/gatk.ug.dbf.%s.filter
out_file = '/mnt/gluster/home/sonal.singhal1/LTF/ancestral_allele/ancestral_allele.%s.cs
genome_out1 = '/mnt/gluster/home/sonal.singhal1/ZF/masked_genome/ZF.masked_genome.fa'
genome_out2 = '/mnt/gluster/home/sonal.singhal1/DBF/masked_genome/DBF.masked_genome.fa'
```

Tuesday, 19 August 2014

In the hassle of restarting ZF yesterday, chromosome LG5 got lost along the way, so I had to restart it from scratch.

I also identified the ancestral alleles for ZF using the following files.

```
vcf_in = '/mnt/gluster/home/sonal.singhal1/ZF/after_vqsr/by_chr/unrel_vcf/gatk.ug.unrel_z
vcf_out1 = '/mnt/gluster/home/sonal.singhal1/LTF/after_vqsr/by_chr/gatk.ug.ltf.%s.filtere
vcf_out2 = '/mnt/gluster/home/sonal.singhal1/DBF/after_vqsr/by_chr/gatk.ug.dbf.%s.filtere
out_file = '/mnt/gluster/home/sonal.singhal1/ZF/ancestral_allele/ancestral_allele.%s.csv
genome_out1 = '/mnt/gluster/home/sonal.singhal1/LTF/masked_genome/LTF.masked_genome.fa'
genome_out2 = '/mnt/gluster/home/sonal.singhal1/DBF/masked_genome/DBF.masked_genome.fa'
```

I started investigating why ZF chr28 ShapeIt Assemble was failing, writing a script that runs it at a range of rhos and window sizes to see if that will get it to work.

Wednesday, 20 August 2014

I wrote a script that takes the ancestral allele info and outputs it in LDhelmet format, which is pretty simple except it did show that I didn't define ancestral alleles at all sites. Looks like the last few sites in a few files never got named. I think it is because I forgot a return statement in one of my functions (previous work has shown that can lead to wonkiness).

Turns out that I could get ZF chr28 ShapeIt assemble to work with rho=0.01 and window size=0.5, so chr28_haplotypes.haps was found using these parameters.

I also had troubles with ZF chr27 ShapeIt assemble, so I tried removing chunks of sites to identify the chunk of SNPs leading to problems (these scripts are all kept under `chr27mystery_*.py`). In the end, I had to remove all SNPs from chromosome position 3090467 to 3251133 in order to get it to run. this subset would not run under a wide range of rhos, either, so it might just be a missing chunk of the recombination map. Olivier Deleneau has the subset of fussy SNPs and is working on it, theoretically.

Phasing uncertainity for all LTF kept failing for the bigger chromosomes, so I edited the `phasing_uncertainty_multisamples.py` file so that you could specify the number of samples to take in. In doing so, it will require less memory. Right now, it required absurd amounts of memory for the bigger chromosomes.

I then had to recalculate θ, given that I had removed all singletons. So, I counted the number of segregating sites after all singletons were removed.

- LTF: 17145060 segregating sites, sequence length = 886164462
- ZF: 22918790 segregating sites, sequence length = 833787048

For zebrafinch, the total number of segregating sites is: 22918790. The total sequence length is: 833787048 bp. So, the segregating sites measure (S_n) is: $\frac{22918790}{833787048} = 0.02748$. Remember that $\theta = \frac{S_n}{a_n}$, where a_n is $\frac{1}{1+\frac{1}{2}+\frac{1}{3}...+\frac{1}{n-1}}$. Here, $n = 38$ because we have 19 diploid individuals. $a_{38} = 0.2317$, which means that $\theta = 0.1186$. Similarly, for long-tailed finch, $S_n = \frac{17145060}{886164462} = 0.0193$ and $a_{40} = 0.229$, which means that $\theta = 0.0845$.

Thursday, 21 August 2014

It turns out that my `make_seq_for_ldhelmet.py` script failed to account for masked sites, so I fixed that in the script and then remade the haplotype files.

I was finally able to get phasing uncertainity for all LTF to finish after a few more buggy script fixes.

I started to create the diploid sequences needed for PSMC for ZF and LTF (`make_diploidsequence.py`). I am making one for every individual, even though PSMC only runs on one individual at a time. I figured that could be my way of bootstrapping. When I actually run PSMC, I will want to remember that I have two subspecies of LTF, and that could be useful somehow.

I also started running LDhelmet. The first step is to find the haplotype configurations, which I did for all the chromosomes at once. It is hard for me to tell if that is the right way to do this, but it certainly seems like it is. It is really slow with all the chromosomes – much slower than just doing one chromosome at a time. Makes me worry that it might be getting overrun in some way, but the memory is not scaling appreciably. The output file is a bit weird because it looks very similar no matter for what chromosome it is run.

I also explored the next step of the process, which is running the table_gen process for generating the likelihood table. It takes a lot of RAM (9 Gb) and is quite slow, but like LDhat, I should only have to do it the once for all my samples.

I also explored the difference between using the simple_pe comand and not in qsub. It looks like not calling it and just asking for multiple threads works more quickly. That seems weird, to me.

Friday, 22 August 2014

The `find_confs` subroutine of `LDhelmet` was running very slowly in its intial configuration, in which I had it find the configurations for all the chromosomes. In fact, it seemed to be not running at all. Well, I thought about it some more and decided that find_confs must all be about the number of haplotypes and the window size, so would be the same for every chromosome in my dataset. I tested that using about 5 or 6 different chromosomes and compared the resulting configuration files across the chromosomes. In fact, they were all the same. (You can easily compare across files using the command `diff`).

Command used: `echo "/mnt/lustre/home/sonal.singhal1/bin/LDhelmet_v1.6/ldhelmet find_conf`

I then started the `table_gen` procedure, which is very similar to how it works in LDhat, in the sense that it provides likelihood values for given rho values for the given haplotype configuration. I used the recommended binning of rho values that Chan provides in the LDhelmet book, which led to the following command.

```
echo "/mnt/lustre/home/sonal.singhal1/bin/LDhelmet_v1.6/ldhelmet table_gen --num_threads
```

Monday, 25 August 2014

Today, I did a random set of stuff.

1. I finally finished the ShapeIt PIR runs for ZF.
2. I realized that, although I confirmed that all fasta files give the same find_confs file as long as they have the same number of haplotypes, I did not confirm that multiple haplotype files give the same find_confs file as a single one. So, I did some tests and confirmed that works.
3. I killed and restarted the LDhat jobs, because I realized that I was running them with a higher memory command than they needed, and also because I had organized the output files in such a way that made it really hard to keep the files organized and not overflowing my machine. The jobs only take, on average, 360M.
4. I realized that for LDhelmet sites for which I didn't have a defined ancestral site, I was setting the probability of a base as being 0.25 for all four nucleotides, but that doesn't really make sense because they don't exist in equal proportions. So, I calculated the relative frequencies of each nucleotide and used those instead. I redid both the ZF and LTF ancestral sites.

A	278825570	0.29
C	207581703	0.21
T	279023215	0.29
G	207672526	0.21

5. I also generated the fasta files for LDhelmet for ZF, again removing all singleton sites.
6. I also generated the table_gen for ZF, using the following command. `echo "/mnt/lustre/home/sona`
7. I generated the pade table for LTF, using the following command. `echo "/mnt/lustre/home/sonal.`
8. And then finally, I tried to get PSMC going. I had already generated the diploid files, as directed. Then, I turned them into FASTQ files with quality score of 40 at every site, because I had already done the necessary quality filtering. And then I turned them into the PSMC-Fasta format, which counts the number of heterozygotes in a given window chunk. I used window size of 10 and 100 bp, but both failed with some sort of segmentation error fault when I ran the following command: `˜/bin/psmc/psmc -N25 -t15 -r5 -p "`425*2+4+6” -o /mnt/gluster/home/sonal.singhal1/
/mnt/gluster/home/sonal.singhal1/ZF/analysis/PSMC/26462.psmcfa+ I will have to keep playing with it to figure out what is going on.

Tuesday, 26 August 2014

I had two paper reviews due this week, so I took part of today to work on both.

I also ran pade for ZF/ `echo "/mnt/lustre/home/sonal.singhal1/bin/LDhelmet_v1.6/ldhelmet pade`

Need to figure out the right block penalty, which is best done through simulations. But, I have no good parameters for the simulations because I have no block penalty! A bit of a Catch-22. So, I will try with what seems like two reasonable block penalty (5 and 50) and get a few chromosomes worth of data, and work from there.

LDhelmet ancestral allele is in index 0 not index 1! Changed.

I also started some preliminary runs for the MCMC portion of things, which looked something like this. These will be used to get the right block penalty, and then I will run them again, probably with more MCMC chains. `/mnt/lustre/home/sonal.singhal1/bin/LDhelmet_v1.`

Wednesday, 27 August 2014

I spent the first half of the day messing around with the preliminary runs of LDhelmet and trying to figure out what the recombination maps looked like, without going through the fuss of actually plotting them. (Too lazy.) Things looked okay, though I will certainly need to take more time to figure out the right block penalty, because the recombination map is quite different n the birds compared to humans.

I also took some time to create alignments for gene trees, which first entailed creating genomes that reflect all the haplotypes. I had to redo this from the exisitng haplotypes because I needed these to include singletons, too. I also wanted to include G_fortis as an outgroup, so that took some finessing.

Thursday, 28 August 2014

I did more paper reviews today, turning both of them in.

I finally made the gene trees, for which I randomly selected 1000 genomic segments of 1000 bp each from across the autosomes in proportion to the chromosome length. I masked for coverage, but ensured that I only considered genomic segments for which 75% of the segment had sufficient coverage at all 3 ingroups. I included G_fortis as an outgroup. I inferred model choice using `MrAIC.pl`, with AICc as the model picker because the number of parameters was far greater than the number of tips. I then inferred the trees using PHYML. It took over 10 hours on 10 clusters to do the 1000 trees!

Friday, 29 August 2014

I used Liang Liu's programs STAR and STEAC on a random set of 1000 gene trees, each of which included all haplotyepes. It took me a while because these programs only work with an old version of R, and that took a while to figure out. Also, it turns otu that I hadn't kept my code from before (muy sad face), so I had to recreate it. (But, I have it now on my local computer!) Both approaches gave the same species tree. Great; these data are stored on my local computer, under `analysis/species_tree`. I plotted this species tree with the original gene trees, but to make it easier to visualize, I plotted them after reducing each tip to 2 or 4 haplotypes, and then removing the outgroup, and then making them ultrametric. I plotted using DensiTree.

I started `*BEAST` runs using Beast 2.0 on a random set of 50 genes. I used a strict clock to help force along convergence. Might try again with uncorrelated clock to see how that goes, but given that phylogeny isn't central to this work, it might not be crucial. Again, I used G_fortis as the outgroup, and I included all sampled haplotypes (N=40 for LTF, 38 for ZF, 2 for DBF, 2 for the outgroup).

I plotted SFS using the ancestral allele information on my local computer in the `sfs.ipynb` notebook.
I finally finished the phasing uncertainty for ZF! The bigger chromosomes had to be cut down in sample number significantly in order to get things to go.

As it would turn out, most of the RJMCMC runs weren't horribly long, so I will run the main ones with the recommended chain length (3$\times$) and then run some other samples with fewer chain iterations to ensure switch error rate isn't leading to false results.