Lab Notebook

Postdoctoral Research

Sonal Singhal

sonal.singhal1@gmail.com

Beginning 7 July 2014

Contents

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 di
2: coverage within acceptable bounds; variant that is HQ and has evidence of Mendelian d
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. Sur
```

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/unifie
```

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_genoty`
  `/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?

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 ňew 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.bamorc
```

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/pha
```

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.

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/zeb`
   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.

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 p̈aper̈is 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$\times$ 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.

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 accurateës̈timates 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 šwitch 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.cove
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.

Tuesday, 2 September 2014

I am an idiot. I calculated theta wrong. The harmonic sum is **not** divided by 1.

species	sites	SS	seq_length	prop SS	a_n	theta	N_e
ZF	all	48726579	859594837	0.0567	4.202	0.0135	3.37e6
LTF	all	27995773	897015175	0.0312	4.253	0.00734	1.84e6
ZF	no singletons	22918790	833787048	0.02748	4.202	0.00654	NA
LTF	no singletons	17145060	886164462	0.0193	4.253	0.00454	NA

I needed to rerun lk and pade steps for both species with the updated thetas. `echo "/mnt/lustre/home/so`
`echo "/mnt/lustre/home/sonal.singhal1/bin/LDhelmet_v1.6/ldhelmet table_gen --num_threads`
I generated species trees from *BEAST (the result is in my home computer). I could run for longer, because the ESS values are a bit low, but it is fine for now. The species tree with dated nodes is same in topology as the STEAC/STAR tree but it has much shallower nodes for the two LTF subspecies.

I also generated a few recombination map graphs to display the initial recombination maps, all of which are in an iPython notebook on my local computer.

Wednesday, 3 September 2014

I reran pade with the updated thetas. `echo "/mnt/lustre/home/sonal.singhal1/bin/LDhelmet_v1.6/l`
`echo "/mnt/lustre/home/sonal.singhal1/bin/LDhelmet_v1.6/ldhelmet pade --num_threads 12 -`

I wrote a script to identify hotspots, given that the 20kb of flanking sequence on either side has to have 10× lower recombination than the focal sequence. I found a lot, though I am pretty skeptical of most of them. I think this is because there is a lot of variation in overall recombination rate across the chromosome, and to me, that is the bigger story. But, I suppose the true is same for humans.
I continued work on graphing recombination, making graphs of the smoothed recombination rate across chromosomes and a graph that showed how proportion of genetic length changed with proportion of sequence. Also, it turned out that I had never graphed the final results from the phasing uncertainty, so I graphed those too. There is a lot more uncertainty than I would like – on average, 5% or so. I also looked to see if there was any correlation between uncertainty and recombination rate, and there was – the two values are correlated around $r = 0.22$ which is (of course) significant given the huge number of points included. I am somewhat concerned about this – Molly had mentioned masking any genomic regions in which uncertainty is too high, but we might get rid of everything interesting if we do that. It could also be that uncertainty is correlated with SNP density (even though I divided by that), which could explain these patterns.

Thursday, 4 September 2014

I am loosely collaborating with Allison Lansverk and Chris Balakrishnan, both of whom are at NCSU, on their work looking at zebrafinch in two ways (1) signals of domestication and (2) what is going on the with the Lesser Sunday zebrafinch. Their sampling design is interesting – they only sampled 3× for each individual, so genotype calling is going to be tricky (they only sampled 20 individuals). Anyways, I spent most of my morning summarizing across the pipelines we used for them. Oh! The other tricky thing they did is that they sampled some museum skins. Uf. These data won't be easy.

I then spent some time researching PSMC, to figure out what sort of plots look normal and the sorts of data and parameter values people use when running them. I had heard that PSMC plots all tend to look the same, but I didn't see that at all.

I was having a lot of trouble figuring why PSMC wouldn't run, so I tried some iterative PSMC runs to see what would work. It turns out that everything is fine, but I just couldn't include the Zchromosome.

In my initial runs, the number of recombinations (¡20) was too low for iterations 57/58 - 63. I will have to mess around with that for the next set of runs.

I also wrote a script to generate fasta files for the block penalty simulations, but they were way too slow. So, I will need to switch to MAcS despite the concerns with this program.

Friday, 5 September 2014

In preparing for my simulations, I figured out that the stationary frequencies for alleles was a bit off (I hadn't accounted properly for mutational biases). These are the new values, which I changed to be the default priors for unknown bases in LDhelmet.

A	0.303
C	0.197
T	0.305
G	0.195

I spent the rest of the day working on the simulations. It took a while to get them going because it is a pretty complicated set of commands, but it looks to be working now. I generated the fasta files and the ancestral allele files through the script, and then I generated the conf/lk/pade files through the command line sung the following commands.

```
echo "/mnt/lustre/home/sonal.singhal1/bin/LDhelmet_v1.6/ldhelmet table_gen --num_threads
echo "/mnt/lustre/home/sonal.singhal1/bin/LDhelmet_v1.6/ldhelmet pade --num_threads 12 -
```

Monday, 8 September 2014

I started simulation runs on LDhelmet, and I wrote scripts to run `post_to_text` and to compare actual and estimated recombination.
Initial psmc runs had no recombination events in least intervals 59 - 64 –also looked like could get more info in near present, so trying with more free parameters.

```
4+25*2+4+6
4+25*2+5
27*2+5+5
27*2+5
4*1+25*2+5
4*1+25*2
30*2
25*2+5+5
25*2+5
2+25*2+2
```

In the mean time, I removed Z chromosome from PSMC files and I created PSMC files with a window of 50.

I spent all afternoon trying to compile msmc – it was tricky because it uses D for compiling. I failed and failed and failed, and then, it turns out that it was already in `/data/tools`.

Tuesday, 9 September 2014

I explored the other PSMC results.

```
4+25*2+4+6 - 63 insufficient
4+25*2+5 - 58 insufficient
27*2+5+5 - 64 insufficent
27*2+5 - 58 insufficient
4*1+25*2+5 - 58 insufficient
4*1+25*2 - 53 insufficient
30*2 - 59 insufficient
25*2 + 5 + 5 - 59 insufficient
25*2 + 5  - 54 insufficient
2 + 25*2 + 2 - 53 insufficient
```

It turns out that having more free parameters close to the front leads to huge inferred population sizes. So, I went ahead and used the suggested one, even though it leads to insufifcient number of recombinations in the last interval. These results also look a little weird, but I will talk to Molly about them. I went ahead and ran PSMC for all ZF and LTF. Now need to plot.
I also made files for treemix. It took some work because Iwanted to split up LTF. Those are made, though they are huge, so I will probably subsample. I also created them with and without DBF, because initial runs showed that including DBF resulted in some funky results – maybe because DBF only has $N = 2$?
I looked at the simulation results. Higher block penalty (500) did better in low recombination areas and lower block penalty (10 or 50) did better in high recombination areas. 100 did fairly well in all, so will go with that. I also reorganized the simulation files so that things look neater, but now my scripts won't work out of the box.

Wednesday, 10 September 2014

I killed the rjmcmc jobs and started again to do species one at a time, and to use just 9 processors at a time. (Otherwise, my runs take up all my available cluster space).

I ran PSMC for DBF.

I ran TreeMix for these populations.

```
/data/tools/treemix/treemix -i treemix_auts_inclDBF.txt.gz -o treemix_auts_inclDBF_tree -
/data/tools/treemix/treemix -i treemix_auts_woDBF.txt.gz -o treemix_auts_woDBF_tree -k 10
/data/tools/treemix/treemix -i treemix_auts_inclDBF_sub.txt.gz -o treemix_auts_inclDBF_su
/data/tools/treemix/treemix -i treemix_auts_woDBF_sub.txt.gz -o treemix_auts_woDBF_sub_tr
```

I continued my work on MSMC.

Thursday, 11, September 2014

I plotted the TreeMix and PSMC results and showed to Molly. Molly suggested the Treemix results should be explored using the 4-tip test and suggested that the PSMC results were off. She wanted me to generate them again with block size of 10.

I had to remake MSMC files because they need to be by chromosome and because I forgot to account for masked sites. Even with the right format, they still kept failing. I will try again with fewer chromosomes.

To figure out mu, there are two ways.

1. I can use the number of cell divisions in a bird between sperm and zygote and egg and zygote, and multiply by 10^{-10} to get an approximate number of divisions. In birds, the number of cell divisions is 32 for males, so that would give 3.2e-9.

2. I can take the relationship between θ and ρ, and divide to find μ given c. In this case, θ in the ZF is 0.0135. c, based on Ellegren's work, is somewhere between 1.3 cM/MB - 1.43 cM/MB, or 1.3e-8 to 1.4e-8. ρ is to be determined.

Friday, 12, September 2014

Need to downsample the MSMC data so that it will actually run. Do not want to subsample randomly because the program uses linkage info, so I will try running with chromosomes.

- LTF – 13e6 SNPs across all.
  - 5% : 650,000 : chr10 + chr11, chr7 + chr8
  - 10%: 1.3e6 : chr1, chr1A + chr6
  - 20%: 2.6e6 : chr1 + chr3, chr2 + chr5
  - 30%: 3.9e6 : chr1 + chr1A + chr3, chr3 + chr4 + chr5
- ZF – 18e6 SNPs across all
  - 5%: 900,000 : chr10 + chr11, chr13 + chr14 + chr17
  - 10%: 1.8e6 : chr1A + chr12, chr1
  - 20%: 3.6e6 : chr2, chr4 + chr5
  - 30%: 5.4e6 : chr1 + chr1A + chr4, chr2 + chr3

This results in the following commands.

```
/data/tools/msmc -o LTF10_11.msmc -t 12 -P 0,0,0,0,1,1,1,1 --fixedRecombination /mnt/glus
/data/tools/msmc -o LTF7_8.msmc -t 12 -P 0,0,0,0,1,1,1,1 --fixedRecombination /mnt/gluste
/data/tools/msmc -o LTF1.msmc -t 12 -P 0,0,0,0,1,1,1,1 --fixedRecombination /mnt/gluster/
/data/tools/msmc -o LTF1A_6.msmc -t 12 -P 0,0,0,0,1,1,1,1 --fixedRecombination /mnt/glust
/data/tools/msmc -o LTF1_3.msmc -t 12 -P 0,0,0,0,1,1,1,1 --fixedRecombination /mnt/gluste
/data/tools/msmc -o LTF2_5.msmc -t 12 -P 0,0,0,0,1,1,1,1 --fixedRecombination /mnt/gluste
/data/tools/msmc -o LTF1_1A_3.msmc -t 12 -P 0,0,0,0,1,1,1,1 --fixedRecombination /mnt/glu
/data/tools/msmc -o LTF3_4_5.msmc -t 12 -P 0,0,0,0,1,1,1,1 --fixedRecombination /mnt/glus
/data/tools/msmc -o ZF10_11.msmc -t 12 -P 0,0,0,0,0,0,0,0 --fixedRecombination /mnt/glust
/data/tools/msmc -o ZF13_14_17.msmc -t 12 -P 0,0,0,0,0,0,0,0 --fixedRecombination /mnt/gl
/data/tools/msmc -o ZF1A_12.msmc -t 12 -P 0,0,0,0,0,0,0,0 --fixedRecombination /mnt/glust
/data/tools/msmc -o ZF1.msmc -t 12 -P 0,0,0,0,0,0,0,0 --fixedRecombination /mnt/gluster/l
/data/tools/msmc -o ZF2.msmc -t 12 -P 0,0,0,0,0,0,0,0 --fixedRecombination /mnt/gluster/l
/data/tools/msmc -o ZF4_5.msmc -t 12 -P 0,0,0,0,0,0,0,0 --fixedRecombination /mnt/gluster
/data/tools/msmc -o ZF1_1A_4.msmc -t 12 -P 0,0,0,0,0,0,0,0 --fixedRecombination /mnt/glus
/data/tools/msmc -o ZF2_3.msmc -t 12 -P 0,0,0,0,0,0,0,0 --fixedRecombination /mnt/gluster
```

Friday, 19, September 2014

I was on vacation for the last four days and was delayed because spudhead was down for most of the day.

I worked on MSMC again, make files again with fewer haplotypes (4) and will run again.

Week of 22 - 26 September 2014

- PSMC: all the runs finished during my vacation with the "window size" of 10, so I took them all and generated the plot. The LTF and ZF lineages coalesce now, but DBF still doesn't make it. However, the divergence times (using my makeshift estimate of μ) look good and make sense, and you can see some evidence for variation between hecki and acuticauda ... so this all seems good.

- MSMC: the reduced sampling worked, but not for all SNP sets. But, when I plotted the initial results (using msmc.ipynb on my local computer), things looked really odd and not what I would have thought. They were more different than I would have thought, and it didn't match up with PSMC, and they "coalesced" much earlier than I would have thought. All in all, nothing much made sense. There are three possibilities here.

  1. My data are phased incorrectly. That will add error.
  2. MSMC sucks. This could be as their simulations are kind of limited, and they don't do any bootstrapping.
  3. MSMC is the wrong program for my data. I get the sense that it is meant to capture more recent population history, but frankly, I do not understand the methods enough to be sure.

  Anyways, despite all this uncertainty, I went ahead and generated multiple MSMC files for each SNP set that worked, so that I can create effective bootstraps, both across comparable SNP sets and across different SNP sets. If things aren't congruent, I am not sure what to do...

- CoalHMM seemed like a good program to look across the three lineages and figure out how they are related, but Molly said no ... after I wasted a ton of time getting the file made and spending tons of time trying to install it. I wonder why it is not good. Is it because it isn't out of the power trifecta?

- TreeMix: I started running the 4 pop admixture test, to see if any of our populations are admixed. It is running very very slowly. The command I used was
  ```
  /mnt/lustre/home/sonal.singhal1/bin/treemix-1.12/fourpop -i /mnt/gluster/home/sonal.
  ```
  I think the reason that it is so slow is because I have two many variants for that k. Maybe I should have increased k? If it doesn't finish in the next few days, I will.

- SNAPP: I spent more time than I would like to admit trying to get SNAPP files created. There were a number of problems, mainly that the BEAST manual is pretty vague about the input format. It turns out that FASTA files are not supported, that 0/1/2 format for alleles is not supported, and it isn't clear that IUPAC ambiguity codes are interpreted as one would hope. So, I created a gazillion and a half files, but I think that I now have the right script (`make_snapp_files.py`) and the right file that will give the results I expect. I ended up subsampling both at the haplotype and the locus level because of everyone's warnings that this program is really really slow.

- Admixture: I created the input files for Admixture, both for all 4 lineages and just for LTF. I tried a number of subsampling regimes, but I ultimately landed upon a sampling scheme that made sense and gave about 300K SNPs. Along the way, I did it removing singletons, but Molly said no, so I am redoing the analyses to not include singletons. The results (on my personal computer under `/Users/singhal/zebrafinch/analysis/admixture/`) didn't make too much sense, because the LTFs didn't break up that cleanly. I guess they really aren't that different, especially when you don't include the Z chromosome. Anyways, those analyses are running again, and I don't know about how to choose K with this program – people do not seem to trust the CV approach. I am using the scripts `make_admixture_files.py` and `make_admixture_files_LTF.py`.
- Simulations: This was the most frustrating thing. Molly didn't like the approach I used for the simulations to identify the right block penalty, so I ended up using the simulation script I wrote for identifying power to detect hotspots instead. This was a non-trivial amount of work, because I started with one approach (where just one hotspot was in the sequence), but it didn't work well because it created way too many files to analyze and it was impossible to identify the maximum number of configurations ... and there was no way that I was going to create a likelihood and pade file for each file. So, instead, I put a number of hotspots into each sequence file, and I am going to evaluate LDhelmet's ability to find hotspots on these metrics: false positive, false negative, rate, accuracy. I will identify the right bpen that way for finding hotspots. This might not be the right bpen for getting the recombination map, so I think that I will end up running LDhelmet at two different bpens. I think that I will end up getting the low bpen is the best, and then I will have too much noise. Oh well.
- Dadi: This is going to be the program that I will use to infer lineage divergence. Annoyingly, they have changed the scripts since I used it last, so I will have to rewrite all my PhD scripts. Installing it was tricky – turns out I should put my Python libraries in the .local drive in my home drive on the cluster. Wrote a script to make the dadi files called `make_dadi_files.py`
- Meetings with Molly: Sucked. Have to redo simulations and then get the analyses started again. Will be giving lab meeting in November. Alva is still going to work on these data for a while. Still not clear what I own versus what Ellen owns. The PAR stuff is legit, like I thought, and like they didn't think. The guy who was going to run LDhat disappeared.

- Other: Got Platypus and GATK implemented for Encelia, but need to make sure that GATK is working. Did a paper review for Current Zoology. Prepared for lab meeting on Monday. Read a lot of papers. Spent half of Wednesday sick.

Week of 29 September - 3 October 2014

- MSMC: Still running. Made no progress on it.
- TreeMix: It was running too slowly because the block size for identifying error was much too small given the size of my data set. Reran as: `/mnt/lustre/home/sonal.singhal1/bin`
  Got results:

  ```
  npop:4 nsnp:78753025
  Estimating covariance matrix in 1575 blocks of size 50000
  Estimating f_4 in 1575 blocks of size 50000
  total_nsnp 78752747 nsnp 78753025
  # columns - f4 stat, SE in f4 , Z-score
  pop1,pop2;pop3,pop4 -0.0439553 0.00018652 -235.659
  pop1,pop3;pop2,pop4 -0.0439529 0.000186517 -235.65
  pop1,pop4;pop2,pop3 2.40322e-06 1.62195e-06 1.48168
  ```

  ... where pop1: zf, pop2: ltfh, pop3: ltfa, pop4: dbf. The interpretation of this is the following – the presumed unrooted tree is ((pop1, pop4),(pop2,pop3)) – pretty straightforward because of course the longtailed finches are the most closely related. So, we want to see how much the realized data fits the treeness of that tree. Here, the z-stat is 1.48 (distance from 0 means worse fit), which is non-significant (p=0.07), so the data do a pretty good job of fitting the presumed tree structure, which suggests that the admixture treemix picked up is probably non-significant.
- SNAPP: I started the SNAPP analyses, but I had to do it on the MVZ cluster because it wasn't working on the Chicago cluster and I couldn't be bothered to see why. `~/bin/BEAST/bin/beast -working -overwrite -threads 12 -prefix snapp_priors all_`
  `~/bin/BEAST/bin/beast -working -overwrite -threads 12 -prefix snapp all_species.snap`
- Admixture: I redid these results with singletons included, and wow, nothing changed.
- Simulations: The hotspot simulations showed the following:
  1. false positives are rare unless you are looking for very small changes in rate (¡10)
  2. very little power to detect things that are less than 2000 bp
  3. tends to underestimate both background rate and hotspot rate – could this be because of singletons? (running simulations to know for sure)

4. very narrow range in which hotspot detection works – from rho / bp from 0.001 to 0.1. above and below that, looks like there is limited power, even when the value of the heat increases a lot.
5. surprisingly, really high heat does not necessarily mean higher chance of detection.
6. only using really high block penalties (100 or 500) leads to loss in finding stuff. but given that false positive rate is so low, might as well go with a low block penalty. 5 and 10 give marginally similar results, so will go with 10.
7. in general, these simulations do not bode well for our ability to find real hotspots, though it turns out these values are pretty similar to other power analyses folks have done.
8. getting general rate is best done with 100.
9. Does suggest that we shouldn't trust results from chromosomes with high recombination rates, because LDhelmet isn't very sensitive to it.

I also became convicned that maybe removing singletons is why the program was performing poorly, so I reran simulations with singletons. `echo "/mnt/lustre/home/sonal.singhal`
`echo "/mnt/lustre/home/sonal.singhal1/bin/LDhelmet_v1.6/ldhelmet pade --num_threads`
Nope, the results still sucked.

I also made graphs, which are in `hotspot_power.ipynb` on my local disk.

I am also scaling up the power analyses (only to evaluate with bpen=5) at Molly's bequest.

- Dadi: The script crashed due to memory, so I reran it and then combined files. Then, I noticed that a given triplet will often have two mutations, which seemed problematic. So, I wrote a short script `prune_dadi.py` that prunes the dadi files so that any mutations less than 3 bp apart are discarded.
- Meeting with Molly: Sucked even more than last time! Have to redo my power simulations. She will support me (modestly) for the NIH. Seemed semi-enthusiastic about the Columbia Science Fellows.
- LDhelmet: Came up with good priors for recombination rate based on the Ellegren data, and started rerunning the files with block penalty of 5 and 100 for all the data. Am using fewer iterations of the MCMC chain so that they will finish some day.
- Other: Read a few papers and finally came up with some hypotheses for my work. Not so clear that I will get funding, though. Lost Tuesday afternoon and most of Wednesday to finding new housing after current roommate turned out to be crazy.

Week of 6 October - 10 October 2014

1. Simulations: listened to Molly but also decided that wouldn't be too informative, so added sims with 10x, 20x, 40x, 60x, 80x, 100x (2 each) too. Finished them and made plots.
2. MSMC: finally finished, and when i made them, there were a few big problems:
   a) population sizes were too big
   b) time mapped was only 500K years rather than the few million years of these lineage histories
   c) there was a lot of inconsistency across the same haplotypes sampled, whether controlled for the species or the chromosome
   d) the population size changes didn't look like PSMC results

   So, i looked into it. there are a few possible issues:
   a) msmc doesn't work for a wide range of histories; some folks say this might true (Priya)
   b) our haplotypes are bad
   c) I had user error

   User error here could be – looks like recombination rate is not estimated (as they suggested) but is automatically set. So, the recombination rate used was almost two orders of magnitude lower than it should have been. So, I used a recombination rate that is hopefully more or less accurate – it was calculated by taking the c estimated from Ellegren (1.3e-8) and multipled by $2 \cdot N_e$. It gave the result of 0.0475, which I used as the recombination rate. Hope that makes things work. Possible error on MSMC's part: they really underestimated the θ, though that could just be because it was a smaller portion of the genome and a subportion of haplotypes was used to estimate theta. Update: new recombination rate decreases population size to match PSMC results better, but the time stamps are still really young.
3. Lab meeting: date for it still unclear. this clarified a need to add more basic population genomics to my work.
4. Other: honestly, spent most of the week on non-zebrafinch work, such as figuring out Encelia and applying for other fellowships.

Week of 13 October - 17 October 2014

1. dadi: made `dadi_make_fs.py` to convert the dadi file into the 2DSFS. It took dadi too much memory to read it in, so I will not be able to use this. What a waste. I will have to turn it into t 2DSFS on my own, actually a 3D SFS on my own. As for possible population histories to fit, I will try migration, no migration, pop growth, and no pop growth, which gives a total of 4 possibilities to code and fit.

2. Simulations: Molly finally okay with simulations, but she wonders if, instead of absolute rho, what matters more is the balance between theta and rho. (Regions of the genome with low rho also tend to have low theta.) So, I first estimated Watterson's theta across the entire genome in 50K chunks. Turns out the range of theta values is pretty low, varying from 0.0073 (10%) to 0.02 (90%). So, I simulated data at theta 0.0075, 0.014, and 0.02 for a range of rhos (all relative to theta). I doubt it will make much difference, but onward and upward. Also, had to figure out what theta was without all those singletons, so based on the simulated data, this is what I got:

   a) theta 0.0075 becomes 0.0057

   b) theta 0.014 becomes 0.0109

   c) theta 0.02 becomes 0.0148

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

3. Other: honestly, spent most of the week on non-zebrafinch work, such as figuring out Encelia and applying for other fellowships.

Week of 20 October - 24 October 2014

1. LDhemlet: LTF - chr16 for bpen5 just kept failing - will need to revisit
2. SNAPP: runs failed; none of the values updated along the chain?! Why / how could that be?
3. Simulations: theta - rho interplay is really hard to disentangle, but it does appear that the ratio of the two values matters as well as rho and theta themselves. So, probably need to filter accordingly.
   a) clearly, anything above 0.5 background rho will have no worth
   b) at really low background rates, having lower theta actually helps, so that's good because that is likely a biological reality
4. Finding putative hotspots: Wrote `find_hotspots_parse.py`.
   a) bpen = 5
   b) heat = 10; rho ratio for spot to background to call a spot
   c) no masking for background rho or theta
   d) includes Zchromosome, which I have to redo, starting with SNP finding
   e) window size = 5K bp; drops any hotspots that are 5K within the last found hotspot
   f) overlap = 10K bp; considers any hotspots that are 10K within each other between the two species to be "matches"
   g) hotspot length = [1000, 2000]; ran the hotspot finder for two sizes for putative hotspots
   h) flank size = 20K bp on either side
   i)
```
species  spot_size  match_in_other_species
LTF      1000       False                     2460
                    True                       464
         2000       False                     1165
                    True                       190
ZF       1000       False                     1276
                    True                       464
         2000       False                      727
                    True                       190
```

5. I wrote the calculate pi script, which is super slow, but at least it let me revisit the idea of pi and segregating sites, and how they connect to Tajima's D. Super sad that I had a better intuition of this when I was an undergrad!

6. Also figured out what is going on with the reference genome for Taegut. The T. guttata genome has been pretty poorly named and organized across sequencing consortiums.

   a) There have been two releases, one of which is called taeGut1 or WUGSC 3.2.4. This was released in 2008.

   b) There is another, which is called taeGut2 or taeGut324, or sometimes as taeGut3.2.4. This was released in Feb 2013.

   c) I have reason to believe that sequencing consortiums are sloppy with the namings, and aren't always using the right name. Check when the genome was released to know for sure.

   d) This project uses taeGut1.

   e) However, there is no clear difference between taeGut1 and taeGut2 in sequence length or sequence identity.

   f) Therefore, the annotation files for either / or should be good.

   g) taeGut1 is still available through the NCBI and UCSC.

7. I also got super excited about pursuing the idea of divergence, polymorphism, and recombination in the long-tailed finch. If nothing else, it could be a great little story to include in my UNH talk, and would provide three really cute (and different) vignettes into the genomics of speciation.

8. Worked only 3 days because it was Manan's birthday / mundun / Diwali. Also did spend a decent amount of time on the fellowships.

Week of 27 October - 31 October 2014

1. Lost Monday to moving and some of Friday because of fellowships.
2. **Finding putative hotspots**:
   a) 40k flanks better than 20k in terns of power, though I am still skeptical. These results are in the `hotspot_power.ipynb` in my local drive.
   b) Not matching hotspots are often still matched, but the other species hotspot is fainter, so not necessarily in the $10\times$ range. Will clearly need to test reciprocally for hotspots, though Molly said there is a more clever way done by Crawford et al. that might be worth exploring.
   c) Some hotspots are clearly larger than the allowed hotspot – need to concatenate these to get the true hotspot length.
   d) Did hotspot finding for 500, 1000, 2000 bp and 20K and 40K flanks. There has to be a better way to do this than trying to do it all iteratively.
3. **Repeat Masking**
   a) We should have repeat masked from the get go. Oh no. Oh well.
   b) http://www.repeatmasker.org/species/taeGut.html
   c) taeGut1 - Jul 2008 - RepeatMasker open-4.0.5 - Repeat Library 20140131
   d) Repeat masked the VCFs and reran biallelic etc accordingly.
   e) Repeat masked the post-text LDhelmet files. This is clearly a subpar solution, but eh.
   f) Created repeat masked genomes.
4. **Pop-gen**
   a) Calculated tajima's D. as expected, strong signal of population expansion
   b) Will need to recalculate using pi that accounts for repeat masking.
5. **Phasing**
   a) LTF does have a higher switch error rate than ZF; median error is 3.21% in ZF compared to 4.496% in LTF
   b) Results are in `switch\_error\_rate.ipynb`.

Week of 3 November - 7 November 2014

1. **Long chromosomes**: Need to have some principled way to define long chromosomes for the analyses, so I calculated mean and median rhos for each chromosome. Most of the tricky chromosomes had median rhos in the range of 0.1 to 1, which is where I did no simulations. So, I needed to figure out how much power I had, if any, in that range. So, I dis some more simulations, adding them to the `sim_10_20_40_60_80_100` simulations on the cluster. I tested rhos from 0.1 to 1, and found the breaking point was somewhere between 0.6 and 0.8.

chr	mean	median
chrZ	0.007	0.000
chr13	0.108	0.060
chr7	0.247	0.076
chr6	0.327	0.067
chr1A	0.151	0.015
chr4	0.107	0.013
chr3	0.094	0.006
chr2	0.076	0.012
chr1	0.123	0.009
chr8	0.292	0.098
chr5	0.163	0.025
chr11	0.263	0.050
chr12	0.491	0.348
chr9	0.389	0.200
chr10	0.425	0.215
chr15	0.743	0.503
chr14	0.491	0.419
chr4A	0.587	0.447
chr27	0.701	0.431
chr17	0.968	0.892
chr16	3.541	3.541
chr19	0.929	0.941
chr18	0.835	0.770
chrLG2	0.764	0.772
chrLG5	9.232	9.232
chrLGE22	0.995	0.372
chr1B	1.472	1.447
chr26	1.058	0.933
chr24	1.343	1.491
chr25	1.891	2.002
chr22	1.352	1.573
chr23	1.135	1.357
chr20	0.833	0.793
chr21	1.196	1.233
chr28	1.224	1.017

Simulations suggested that power drops off around 0.8. So, I will use chromsomes 1 - 15 (incl. 1A, 4A but not 1B) and chr27 and chrZ.

```
longchrs = [  'chr1', 'chr1A', 'chr2', 'chr3', 'chr4', 'chr4A', \
'chr5', 'chr6', 'chr7', 'chr8', 'chr9', 'chr10', \
'chr11', 'chr12', 'chr13', 'chr14', 'chr15', 'chr27', 'chrZ']
```

Also recalculated average rho for these chromosomes only. For ZF, median rho is median rho: 0.02012 (in units per bp). For LTF: median rho: 0.01207 (in units per bp)

2. **TSS**: Looked at the rho around TSS and plotted on home computer under `/Users/singhal/zebrafin` Looks somewhat similar to human and dog data, in that we see a peak of rho near the TSS, but it is not structured the same way.

3. **95 HPD useful?**: Looking at if the 95% HPD is informative. To do this, looked at map simulations and saw how often actual rho estimate falls within inferred 95 HPD. Only 325035 out 1914723 of the sampled points fall within the distribution, which is 17%. (Results on `/mnt/gluster/home/sonal.singhal1/ZF/analysis/map_simulations/s` If restricting it to the magic transition point (`rho` < 0.8) this sampling gets more accurate, but is still only 19%. So, I think the 95% HPD tells us very little, so I am not going to pursue it further. The deviation between actual and inferred rho has to be really low for the 95 HPD to count – average deviation was 1.01, compared to 1.23 for those in which the 95 HPD didn't overlap. Clearly, it has to be fairly on point for it to be in the HPD. The HPD seems to be a bit narrowly defined.

4. **Compare rhos in species**: calculated smoothed rho across multiple window sizes (10Kb, 100 KB, 1 MB, 5 MB); wrote `compare_rho_correlations_species.py`; analyzed on `compare_zf_ltf_rho.ipynb` on my personal computer. Looks like correlations are pretty high, especially for long chromosomes ... but that there are a few chromosomes which are doing pretty different things.

5. **Compare possible hotspot finders**: Tested three programs sequenceLDhot and InferRho and PHASE on their abilities to run on a set of randomly selected 20 hotspots. Script is called `find_hotspots_multiple_validation.py`.. It generates both the sequence files and the shell scrip to run them. Initial results: InferRho is too slow. PHASE doesn't seem to be picking any of the hotspots up. SequenceLDhot is picking up some of the hotspots ... but also picking up other hotspots along the way. In the meantime, Jeff Wall at UCSF is running his modified version of LDhot for our data. He needs to generate the likelihood table and we'll go from there. Note that sequenceLDhot only requires 200M of RAM. I eventually killed InferRho because was taking too long to run. I also wrote the script `find_hotspots_validate.py` to run seqldhot for all the hotspots, as most conservatively defined with block length of 2kb and flank of 40Kb.

6. **Recalculated thetas:** Need to recalculate thetas based on repeat-masked data.

7. **Z chromosome:** First, I filtered the Zchromsomes. Then I recoded them so that any heterozygous site in females was seen as wrong (as it is) and was replaced with an N. Any variable position that had 3 or more female hets was considered bad and ditched. Then, I created a VCF with males only, which was all that was phased. (Haploid chromosomes don't need to get phased...)

    a) First ran extractPIRS.

species	af	chr	SS	seq length	SS frac	a_n	theta	N_E
zf	af0	all	43937119	796084847	0.0552	4.202	0.0131	NA
zf	af1	all	21412173	796084847	0.0269	4.202	0.0064	NA
zf	af0	long only	38894800	779429876	0.0499	4.202	0.0119	NA
zf	af1	long only	19039396	779429876	0.0244	4.202	0.0058	NA
ltf	af0	all	25735433	832034824	0.0309	4.253	0.0073	NA
ltf	af1	all	16018144	832034824	0.0193	4.253	0.0045	NA
ltf	af0	long only	22478765	708345822	0.0317	4.253	0.0075	NA
ltf	af1	long only	13914413	708345822	0.0196	4.253	0.0046	NA

```
~/bin/extractPIRs.v1.r68.x86_64/extractPIRs --bam /mnt/gluster/home/sonal.singha
~/bin/extractPIRs.v1.r68.x86_64/extractPIRs --bam /mnt/gluster/home/sonal.singha
```

b) Then created a reference panel for phasing from the female haploid chromosomes, using `chrZ_make_shapeit_hapref_females.py`.

c) Then I checked the reference against the sites to get phased. Note, a few sites were completely missing in males, so deleted those.

```
missing = [15901058]
/mnt/lustre/home/sonal.singhal1/bin/shapeit_v2r790/shapeit -check --input-vcf /m
missing = [12297573, 25324470, 31785435, 61311164]
/mnt/lustre/home/sonal.singhal1/bin/shapeit_v2r790/shapeit -check --input-vcf /m
```

d) Then I ran ShapeIt, which was lightning fast!

```
/mnt/lustre/home/sonal.singhal1/bin/shapeit_v2r790/shapeit -assemble --input-vcf
/mnt/lustre/home/sonal.singhal1/bin/shapeit_v2r790/shapeit -assemble --input-vcf
```

e) Then, I made chrZ haplotypes without singletons for LDhelmet using `chrZ_make_seq_for_ldhe`

f) then, I recalculated theta for chrZ, because it has a different chromosome number.

```
ZF Z chromosome theta
num SS: 268945
seq length: 61086180
a_n: 3.891
num-chrom: 28
theta: 0.00113

LTF Z theta:
num SS: 412277
seq length: 65344460
a_n: 4.027
num_chrom: 32
theta: 0.00157
```

g) Then, made ancestral alleles for chrZ again. No change to the script.

h) Then ran all the LDhelmet nonsense with the new thetas. Interestingly, there were a lot more confs for these chromosomes. I wonder why? I hope that doesn't reflect poor phasing, though I don't see how it would.

```
~/bin/LDhelmet_v1.6/ldhelmet find_confs -w 50 --num_threads 8 -o /mnt/gluster/ho
~/bin/LDhelmet_v1.6/ldhelmet find_confs -w 50 --num_threads 8 -o /mnt/gluster/ho
/mnt/lustre/home/sonal.singhal1/bin/LDhelmet_v1.6/ldhelmet table_gen --num_threa
/mnt/lustre/home/sonal.singhal1/bin/LDhelmet_v1.6/ldhelmet table_gen --num_threa
/mnt/lustre/home/sonal.singhal1/bin/LDhelmet_v1.6/ldhelmet pade --num_threads 12
/mnt/lustre/home/sonal.singhal1/bin/LDhelmet_v1.6/ldhelmet pade --num_threads 12
/mnt/lustre/home/sonal.singhal1/bin/LDhelmet_v1.6/ldhelmet rjmcmc --num_threads
/mnt/lustre/home/sonal.singhal1/bin/LDhelmet_v1.6/ldhelmet rjmcmc --num_threads
/mnt/lustre/home/sonal.singhal1/bin/LDhelmet_v1.6/ldhelmet rjmcmc --num_threads
/mnt/lustre/home/sonal.singhal1/bin/LDhelmet_v1.6/ldhelmet rjmcmc --num_threads
```

8. **Phased haplotype files:** I'm an idiot – the problem with using `biallelic_SNPs*vcf` for cross-species comparisons is that these files didn't contain fixed SNPs. These are the files I used for a number of things, so I need to redo those analyses (*BEAST, TreeMix, GeneTrees, STEAC/STAR). So, to do that, I first created phased VCF files using the haps data with the script `phase_vcfs_using_haps.py` and `chrZ_phase_vcfs_using_haps.` Then, I created the haplotypes again using `make_haplotypes.py`, with some modifications to account for the diffeent data source.

9. **Random Notes:** Remember that in *ids.txt file, 1 = female and 2 = male. This is backwards from how most programs identify it.

Week of 10 November - 14 November 2014

- Had to make ZF chromosome 2 haplotypes again, failed for some reason.
- Ran post-to-text for chrZ LDhelmet for both species.
- Found hotspots on Z chromosome for ZF; no modifications to script.
- Estimated that mu for ZF is about 2.7e-9.
  1. theta of long chromosomes = 0.0119
  2. median c of long chromosomes = 0.1284e-8 (because of 1e6 for Mb and 1e2 because a percentage) (calculated by using `~/scripts/simple_median_c_rate.py`)
  3. median rho of long chromosomes = 0.02497 (calculated using `simple_summarize_rho_rates.py`
  4. $\frac{\mu}{c} = \frac{\theta}{\rho} = \frac{\theta \cdot c}{\rho} = \frac{0.0119 \cdot 0.1284 \cdot 10e-8}{0.02497} = 6.12 \cdot 10^{-10}$
- Smoothed windows for Z chromosome
- Fixed seqLDhot script to use VCFs, so it is more flexible. Than ran seqLDhot for ZF for all hotspots. Will start LTF and DBF soon.
- Redid phase uncertainty for Z chromosome.
- Wrote first draft of methods.
- Redid STAR / STEAC.
- Replotted gene trees.
- Initial results suggest 539 out of 704 hotspots are confirmed!; results from 'round1' and plots are in `analysis/hotspot_validation`
- Redid treemix to include fixed sites.
- Redid STARBEAST for 3 sets of 50 loci each, using relaxed lognormal clock with mutation rate 0.0027 per site in million years. Used fixed substitution and clock model across loci, because even though not true, keeps things running.

Week of 17 November - 21 November 2014

1. Fixed TSS so that the orientation of genes on the reverse strand are also 5′ to 3′.
2. Finished rerunning Treemix.

   ```
   npop:4 nsnp:76608539
   Estimating covariance matrix in 1532 blocks of size 50000
   Estimating f_4 in 1532 blocks of size 50000
   total_nsnp 76608539 nsnp 76608539
   pop1,pop2;pop3,pop4 -0.0488578 0.00064788 -75.4119
   pop1,pop3;pop2,pop4 -0.0488563 0.000647846 -75.4135
   pop1,pop4;pop2,pop3 1.56294e-06 1.58227e-06 0.987782
   ```

   Here, the z-stat is 0.98 (distance from 0 means worse fit), which is non-significant (p=0.32), so the data do a pretty good job of fitting the presumed tree structure.
3. Updated Methods, shared with Alva and Ellen
4. BAH. Messed up mu calculations. corrected. The new estimate is $6.1 \cdot 10^{-10}$, which is really low. It gives a N_e estimate that makes sense, but ...
5. Started redoing the recombination graphs. Too many of these chromosomes don't make it to 50 cM, which is what you'd expect if you get 2 crossovers per chromosome in every individual, which is what they argue you need for proper segregation during meiosis. It is quite possible that I have misestimated Ne, but, I also think that the mutation rate between ZF and LTF would have to be quite different for this to work for both. Maybe I misestimated θ for LTF? How could that be?
6. Started making files for hotspot power simulations. These needed to be scaled up, so am doing enough simulations per parameter set to get 100 hotspots.
7. Created PHASE files for all hotspots – will check if PHASE does better than seqldhot. Am allowing PHASE to rephase and also giving PHASE known phases. Note: I am running MR2, which is different from MR1, because you have to tell it where the hotspot is. This worries me because I think that is an imprecise thing, but so it goes. Having PHASE identify the phasing is going SUPER slowly, so I think that I won't do that for LTF, nor will I do it for more than a few of these hotspots. Will do it just to appease Molly. Will need to run each hotspot 3 times though, because that's what the

8. Substitution * clock rate gives total evoutionary rate, so don't try to estimate both, because things become non-identifiable. This is why my new BEAST runs failed. To fix it, I need to not estimate subsitution rates, and just allow clock rates to vary. I should probably not have one clock rate for all of the data like I did before. And I should probably allow multiple substitution models across the sites, but then there will be way too many parameters...

Week of 24 November - 26 November 2014

1. SequenceLDhot reciprocal; these finally finished
2. Reran PHASE to be full scale for both species
3. Full simulations for power – started them running, 500 simulations in total
4. Rewrote counting of multiallelic sites to be simpler, `count_multiallelic_sites.py` and reran.

   ```
   LTF:
   multiallelic SNPs: 465636
   all SNPs: 26173532
   percent: 1.8%
   ZF
   multiallelic SNPs: 1643215
   all SNPs: 45566004
   percent: 3.6%
   ```

5. Data on wrong female genotypes.

   ```
   LTF
   all_sites: 1907667
   filtered_sites: 30263
   all_female_geno: 15261336
   wrong_female_geno: 233357

   ZF
   all_sites: 826189
   filtered_sites: 15664
   all_female_geno: 8261890
   wrong_female_geno: 154541
   ```

6. Realized that I never calculated the median / mean rates for longtailed, so did that. The values are lower, as expected, but looks like hotspot finding should be fine in all comparable chromosomes but for chrZ (`summarize_rho_chromosomes.py`).
7. `find_hotspots_seqldhot_parse.py` – Gets the data on seqldhot matches.

Week of 1 December - 12 December 2014

1. First, I wrote a script to parse PHASE (`find_hotspots_phase_parse.py`) which found that PHASE validated many fewer hotspots than SeqLDhot. This is likely because (1) PHASE is less powered and (2) PHASE requires one to identify where the hotspot is.
2. Also, to ensure that errors in phasing aren't influencing hotspot detection, I ran PHASE in the mode where it re-estimates haplotypes as well. I then compared these rephased results to the ShapeIt phased results (asphased). The two sets of results are pretty much the same. The script used was `find_hotspots_phase_rephased.py`.
3. I uploaded the phased VCF files and the haplotype data to Jeff on my Columbia Google Drive. I will keep the data there because it seems like a good just-in-case backup.
4. I plotted hotspot validation data in my local notebook `hotspot_validation.ipynb`.
5. Met with Molly and we discussed why the hotspot finding has been so erratic. We came up with a few options.
    a) SNP calls are bad
    b) Phasing is bad
    c) LDhelmet is inaccurate
    d) seqldhot / PHASE are too sensitive

   How to check for these options?
    a) use LDhat because doesn't require phasing
    b) do simulations to see how switch error rate influences FDR
    c) see if hotspots are in areas with low SNP quality, high switch error rate, lots of Mendelian error
    d) do sensitivity test of PHASE
    e) do sensitivity test of seqLDhot
    f) compare family phasing to each other when run as three trios
    g) compare family phasing to population phasing
    h) rephase population using family phasing
    i) look at hotspots under phase uncertainty

6. I rephase ZF population using family phasing.
   a) make HAPI files: no multiallelics,
   b) rerun PIR1
   c) run HAPI: had to be careful with output because will need to change chrom numbers
   d) get HAPI files to be in reference format – mind missing data
   e) rerun PIR2 `/mnt/lustre/home/sonal.singhal1/bin/shapeit_v2r790/shapeit -assemble -`
   f) Rephase population using family phasing (`old_results` and `old_hap` vs. `results/`)
   g) Some ShapeIt phasing failed, in which case I just used the old phasing without the bases in repetitive areas.
7. Finished the full simulations for power. There are fake hotspots! boo. i think some of this may be that my definition of false positive is too sensitive...
8. Helped find bug for Amy in HAPI.
9. I did simulations to see how switch error rate influences FDR; wrote script: `simulations_FDR.py`.
10. With all previous analyses, had been looking only at the most trusted set of hotspots. Let's blow it up, a bit. Basically, want to confirm that all hotspots found in the most lax settings also find all the more conservatively defined hotspots. Looks like most of the matches in the most conservative set are recapitulated in one of the other sets. Only 38 in the most conservative set are unique to this set. Looks like there are 3519 unique hotspots in ZF and 6291 unique hotspots in LTF. So, the general pool of hotspots is much greater than the most conservative set that we tested. Important to think about with respect to identifying how "matching" the two genomes are. Maybe this is the importance of identifying and using coldspots? (`hotspot_issues.ipynb`)
11. Fixed all recombination maps to be in cM assuming same genetic length
12. Ran LDhat on genotype data to see if it works as well as on phased data. Trying to see if phasing is in fact the issue. `~/bin/LDhat_v2.2/complete -n 38 -rhomax 100 -n_pts 101 -t` and generated LDhat files. LDhat seems super underpowered, and also, it recovered hardly any hotspots.
13. Worked on lab meeting presentation.
14. Made the pi to recombination rate plot.
15. Finished recalculating pi, theta, and tajima's D and made pop gen graphs.

Week of 15 December - 19 December 2014

1. Gave lab meeting. Sucked. Whatever.
2. Fixed many figures based on lab feedback.
3. Identified cold spots for Ellen. Turns out these need to be matched for base composition, which I haven't done yet. Script is `identify_coldspots.py`.

```
ANCESTRAL GENOME
/mnt/gluster/home/sonal.singhal1/reference/ancestral_genome.fa
VCF FILES
LTF: /mnt/gluster/home/sonal.singhal1/LTF/after_vqsr/by_chr/*phased*gz
ZF: /mnt/gluster/home/sonal.singhal1/ZF/after_vqsr/by_chr/unrel_vcf/*phased*gz
HOTSPOT DATA
LTF: /mnt/gluster/home/sonal.singhal1/for_ellen/seqldhot_validated_hotspots.LTF.csv
ZF: /mnt/gluster/home/sonal.singhal1/for_ellen/seqldhot_validated_hotspots.ZF.csv
COLDSPOT DATA
LTF: /mnt/gluster/home/sonal.singhal1/for_ellen/ldhelmet_unvalidated_coldspots.LTF.c
ZF: /mnt/gluster/home/sonal.singhal1/for_ellen/ldhelmet_unvalidated_coldspots.ZF.csv
```

4. Made figure 1E from auton 2012 (corr in 5Mb chunks). Is in `recombination_maps.ipynb`. Did this as correlation rather than difference as in Auton 2012 because it made more sense.
5. Will attempt to account for phase uncertainty for hotspots. To do so, wrote script `find_hotspots_phasing_uncertainty.py`. This script generates seqldhot input files based on randomly sampled haplotypes from the haplotype graph. I ran these results and put them in `/mnt/gluster/home/sonal.singhal1/ZF/analysis/hotspots/hotsp`
6. I then rephased the unrelated population using the parental haplotypes from the family as reference. Reran LDhelmet, had to restart two of them, and it looked like there were issues with one of the clusters.
7. In order to get an estimate of switch error rate, I am going to compare phasing of the family as done with hapi versus as done with the PIR approach. To do so, I had to do the standard litany of creating biallelic VCFs, running PIR1, running PIR2, and then getting rid of completely missing sites.

Week of 5 January - 9 January 2015

1. Had to redo ancestral genomes because G. magnirostris and G. fortis genomes weren't quite right. They were too short, which happens because samtools faidx will not call a base N unless it at least has some coverage in it. So, I`geospiza_genomes.py` to help take care of this issue. In the meantime, I had to regenerate the G. magnirostris vcf, using this command. `~/bin/samtools-0.1.19/samtools mpileup -I -uf /mnt/glus`

2. Compared hotspots found with ZF phased using family reference and without family phasing.
   a) putative hotspots for ZF, without fam: 874 – 316 match in both species
   b) putative hotspots for ZF, with fam: 603 – 317 match in both species
   c) gosh darn. Phasing matters so much. I really worry about all these results.
   d) will need to look at these results and decide which phasing set to go with.

3. Using the results from hapi, I compared the haplotypes the children inherited to that they got from their parents. Any differences should be either from CO, NCO, or errors.
   a) way too many breaks (0.0027 out of a heterozygous site)
   b) if you get rid of the small breaks (some of which could be gene conversions) it goes down to 0.00023
   c) many of these must be errors
   d) `recombination_breaks_hapi.py` is the script to do it.
   e) `/mnt/gluster/home/sonal.singhal1/ZF/phasing/family_approach/hapi/all_chr.breaks`

4. Found hotspots for full range for `/mnt/gluster/home/sonal.singhal1/ZF/analysis/LDhelmet/wi`

5. Switch error rate: compare family phased with HAPI to family phased using ShapeIt. Note that you lose some SNPs using the family method. the median switch error rate (plotted in `switch_error.ipynb`) howevers around 1% to 10% for each chromosome. It is really high on chromosome 1 at around 5e7 bp (or halfway through). This doesn't surprise me because chromosome 1 was super hard to phase. I suspect this is an issue with file corruption. Also, looks like switch error rate is higher where recombination is higher. This makes sense, as hapi attempts to minimize recombination at all times, which, if recombination is actually high, would lead to false negatives.

6. Remade graph for switch error rate; in `hotspot_issues.ipynb`. Wwitch error rate doesn't look to be associated with hotspot occurrence. Interesting.

7. Molly wants to get rid of any sites that are in areas of short switches, with the assumption being that we are getting way more than you'd expect just due to NCOs, so they must be errors. I plotted the number of switches based on the length of switch that was ditched, and it looks like if you remove just the shortest of switches (1) I more than half the number of switches. By removing longer switches (up to 10), I get 90% removed. You have to remove really long switches (50K) in order to get 20 or so switches .... which is still too much. So, clearly the data have issues, but you do what you can. I will mask all the sequence within the bad switch. Because I have the repeatmasked sequence and this sequence to remove, I don't want to do it posthoc anymore. That's janky. Molly agrees. So I am sadly starting anew. And I made the bold decision to start from the real beginning because I am concerned about the quality of the pipeline because we have changed our minds so often.

8. Need to decide how to handle LTF – should this sequence be masked for them too?

9. Step 1 of new pipeline: regenerating coverage-masked and repeat-masked VCFs from the raw VCFs for the family and unrelated ZF.

Week of 12 January - 16 January 2015

So, this week was frustrating. I did everything up to running PIR. But then, when I ran ShapeIt in Assemble mode, all the files failed. I think it is because combining the unrelated and family samples lead to a lot of poor quality SNPs being called real SNPs. I can see this in the huge increase of heterozygous sites called on the Z chromosome. To hopefully get things to work again, I am going to start the pipeline again only using the SNPs called in the unrelated zebrafinches. Unfortunately, the cluster was down on Friday and I was at the NYAPG conference Thursday, so that lead to a few lost days.

Week of 20 January - 23 January 2015

1. Taking out sites in the short switch stretches ultimately removed sites as such:
   a) 3.3% of sites overall (though doesn't account for rm, coverage) and is probably a greater proportion of SNPs
   b) 5.3% of each chromosome, on average

   However, removing the switched regions reduced the number of small breaks by 99.4%. Many chromosomes, especially the smaller chomosomes, now have no small breaks. (Calculated using `recombination_breaks_hapi_parse.py`)

2. Redoing the sex chromosomes gave the following error rates.

```
ZF
all_sites: 830723
filtered_sites: 16513 (2\% of sites)
all_female_geno: 9137953
wrong_female_geno: 165422 (1.8\% of geno)
LTF
all_sites: 1907667
filtered_sites: 30263 (1.6\% of sites)
all_female_geno: 15261336
wrong_female_geno: 233357 (1.5\% of sites)
```

3. Redid both steps of the PIR for chrZ.

```
# PIR1
~/bin/extractPIRs.v1.r68.x86_64/extractPIRs --bam /mnt/gluster/home/sonal.singhal1/L
~/bin/extractPIRs.v1.r68.x86_64/extractPIRs --bam /mnt/gluster/home/sonal.singhal1/Z
# assemble files
/mnt/lustre/home/sonal.singhal1/bin/shapeit_v2r790/shapeit -assemble --input-vcf /mn
/mnt/lustre/home/sonal.singhal1/bin/shapeit_v2r790/shapeit -assemble --input-vcf /mn
```

4. Recalculated thetas for use in LDhelmet.

```
Thetas without singletons
ZF
autosomes unmasked sites: 720884816
autosomes non-singleton sites: 20452332
autosomes nchr: 38
```

```
theta: 0.00675

chrZ unmasked sites: 54450385
chrZ non-singleton sites: 663470
chrZ nchr: 28
theta: 0.00313

LTF
autosomes unmasked sites: 773166097
autosomes non-singleton sites: 15526465
autosomes nchr: 40
theta: 0.00472

chrZ unmasked sites: 58868727
chrZ non-singleton sites: 643378
chrZ nchr: 32
theta: 0.00271
```

Week of 26 January - 30 January 2015

1. Regenerated the table gen and pade files.

   ```
   ~/bin/LDhelmet_v1.6/ldhelmet table_gen --num_threads 12 -c /mnt/gluster/home/sonal.s
   ~/bin/LDhelmet_v1.6/ldhelmet table_gen --num_threads 12 -c /mnt/gluster/home/sonal.s
   ~/bin/LDhelmet_v1.6/ldhelmet table_gen --num_threads 12 -c /mnt/gluster/home/sonal.s
   ~/bin/LDhelmet_v1.6/ldhelmet table_gen --num_threads 12 -c /mnt/gluster/home/sonal.s
   ~/bin/LDhelmet_v1.6/ldhelmet pade --num_threads 12 -c /mnt/gluster/home/sonal.singha
   ~/bin/LDhelmet_v1.6/ldhelmet pade --num_threads 12 -c /mnt/gluster/home/sonal.singha
   ~/bin/LDhelmet_v1.6/ldhelmet pade --num_threads 12 -c /mnt/gluster/home/sonal.singha
   ~/bin/LDhelmet_v1.6/ldhelmet pade --num_threads 12 -c /mnt/gluster/home/sonal.singha
   ```

2. Started LDhelmet running again with new haplotypes. Need to do for bpen5 for ZF first, and then for LTF.

3. Checking how sensitive seqLDhot is to background rho, so rerunning each inferred hotspot at background rho of 1.5x and 0.5x respectively, and seeing how that affects seqLDhot's ability to find the hotspot.

4. Checking how sensitive PHASE is to where PHASE is told where the hotspot is. Checked to see if PHASE can find the hotspots that SeqLDhot found, but that PHASE didn't, when given the "right" hotspot coordinates inferred from SeqLDhot: `find_hotspots_phase_check_sensitivity.py`.

5. Had to reinfer graphs for phase uncertainty for hotspots. Need to start this from the beginning due to a serious error in the phase uncertainty for hotspot script. Will redo this with the new haplotypes and the new hotspots.

6. Recalculated switch error rates and plotted in `hotspot_issues.ipynb`. The new switch error rates are much lower.

7. I looked into the inversion calls that Daniel had made in `zebrafinch/analysis/inversions/` and didn't find much of anything. I honestly don't trust his results, because so many of the inversions were so large and were shared between the species. Given a subset of more trustworthy results, I didn't find many patterns except that inversion breakpoints tended to be in areas of low and high recombination. While you'd expect breakpoints to be in areas of high recombination, I don't know why there would be more in low recombination.

8. I generated the SFS for chrZ and autosomes, separately using `1D_sfs.py`. Autosomal SFS looked fine, but the chrZ SFS (although having pretty much the same shape as the autosomal SFS) had some wonky spikes, no obvious reason why. I looked at the subset of sites that fell into the wonky spike category (it was really just one category in each SFS because of the mirroring of the SFS), and there was nothing wrong with them. I worry that it might be because of the filtering I did in recoding for sex.

9. Running PSMC for chrZ, males only. Saw no need to randomly combine haploid chrZ, so didn't. Window size 10.

10. Started to look at where the PAR is. Ellen identified the zebrafinch PAR on 1 - 450000. The flycatcher PAR – first contig is definitely on the same chromosome from 10K to 300K, but the second contig maps largely to chr24 - 100K - 600K. To confirm that chr24 might be the PAR, will need to look at chicken results. Also, while files were mapped to the chrZ_random, no SNPs were called on them, so analyzing the PAR would require doing SNP calling again.

11. SeqLDhot sensitivity: SeqLDhot is not that sensitive to the background rho rate being inferred incorrectly.

Week of 2 February - 6 February 2015

1. Ran PSMC for chrZ – looks super weird in that it is more noisy, but more or less, follows autosomes.

2. This lead me to look at patterns of variation on the Z chromosome (and all the other chromosomes), and I found that the Zchromosome has a very unusual pattern where the telomeres are super differentiated and the middle of the chromosome is hardly differentiated at all. Maybe this is because of the inversions?

3. I found hotspots for LTF and ZF, using all the ranges of block sizes and flank sizes, though we are only going to pursue the most conservative hotspots found with block 2000 and flank 40000.

4. I got samples from graphs for phasing uncertainty, which I won't be able to do until I know what the hotspots are.

5. Check how sensitive PHASE is to location of hotspot: To see how sensitive PHASE is to its "greatest" weakness, that the hotspot location has to be given as a known, I ran PHASE at 415 hotspots that had been previously validated using SequenceLDhot but weren't validated using PHASE. And, rather than giving the hotspot coordinates inferred by LDhelmet, I used those given by SequenceLDhot. 61 (at cutoff 10) passed and 211 out of 415 (at cutoff 5) passed. This is more than what passed the first time (0). Further, the inferred rho is 1.28x (mean) - 1.32x (median) higher than what we gave. These results suggest we might be under inferring the background rate, but our SeqLDhot results suggest that even if when validated hotspots are run at a background rho 1.5x higher than inferred, we still capture many of the hotspots. 121 out of 415 have a re-inferred background rho as being greater than 1.5. It is possible that when the background rho is much higher that it will have a big effect. All these results suggest that the discrepancy between PHASE and SeqLDhot is partially due to PHASE being underpowered and partially because hotspot location needs to be specified, `find_hotspots_phase_check_sensitivity_parse.py`.

6. one of the PAR contigs from flycatcher does match mainly to chr24. Getting data from Qi Zhou at UC Berkeley to look at PAR in ground finch. Ground finch PAR is in scaffold 143. Good match in genes (at least 9, but probably more – annotation qualities are quite poor) across all 3 species (`PAR_compare_genelists.py`). The chr24 seems to be an issue with the quality of the flycatcher genome. So, can trust our PAR.

7. Created seqldhot files to run. Because I am looking at heat ¿ 5, there are many more files to run. 7K each. It will take 2 weeks or so.

8. In how much of the genome do we have rho values (0.001 - 0.6) in which we can actually infer hotspots? (in `sequence_length_rho.py`).

type	length
total genome length	1021462940
long chr length	929937698 (91%)
ZF	733641215 (78.9% or 71.8%)
LTF	825865157 (88.8% or 80.9%)
shared in both ZF and LTF	699449179 (75.2% or 68.5%)