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, mutu

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.v
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_g
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/gat
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_summar
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_ch`
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?

Thursday, 10 July 2014

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

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

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

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

Friday, 11 July 2014

1 Data management.

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

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

2 Phasing variants.

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

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

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

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

because we have 19 diploid individuals. $a_{38} = 0.2317$, which means that $\theta = 0.2447$. $\theta = 4N_e\mu$, and let's take $\mu = 1 \times 10^{-9} \frac{mutations}{site \cdot gen}$, which means $N_e = 61 million$. That's absurd. If mutation rates are more on the order of humans, then we get a more reasonable $N_e = 6 million$. This actually concords quite well with the Balakrishnan and Edwards value of $\theta = 7 million$. 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 gene 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, $\rho = 4 \cdot 6e6 \cdot 1.2e - 8 = 0.312$, and for longtailedfinch, $\rho = 4 \cdot 3.4e6 \cdot 1.2e - 8 = 0.1768$.

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

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

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

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

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

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

Weekend and Monday, 14 July 2014

1 Phasing variants.

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

2 Generating recombination map.

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

Tuesday, 15 July 2014

1 Phasing variants.

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

2 Generating recombination map.

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

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

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

Wednesday, 16 July 2014

1 Phasing variants.

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

2 Generating recombination map.

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

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

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

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.