Ishaan Jhaveri
Instructor: Amy Williams
BTRY 4840/6840, CS 4775 Fall 2016: Computational Genetics and Genomics
14 December, 2016

What can alleles on certain Single Nucleotide Polymorphisms (SNPs) with anomalous p-values calculated using Gene Dropping (GD) techniques tell us about genetic variation in a natural population?

1 Introduction

Genotypes of members of any population over time are influenced by genetic forces. Forces such as inbreeding lead to more homogeneous genotypes, whereas forces such as mutation lead to heterogeneous genotypes. Other genetic forces like natural selection are also at play. An interesting problem would be to calculate and quantify just how much these forces contribute to genetic variation in a population. To this end, Gene Dropping (GD) is a popular technique used by population geneticists to solve this problem. The idea is use computer simulations to create a null hypothesis for what the genotypes of members a population would look like if all individuals bred completely randomly, and all alleles passed from parent to offspring purely following Mendel's Law of Independent Assortment. The null hypothesis for each individual's genotype can then be compared to each individual's actual genotype. Differences in the simulated genotypes and actual genotypes can give insights into the influence of various genetic forces. In order to do this one needs a pedigree of individuals with genotypic data for each individual, as well as information about how each individual is related to the other individuals. In this project, we create a program that can perform GD, and test it on data from an endemic population of Florida Scrub-Jay's consisting of a little over twelve thousand individuals. The genotypic data for these individuals is at the Single Nucleotide Polymorphism level. In the test dataset we have 1262 SNPs for each individual.

1.1 Related Work

Macluer et al (1986) used GD to analyze the extent of genetic variability in a colony of small South American marsupials, *Monodelphis domesticus.* Since then GD programs like Mendel, Merlin, GENLIB, IBDsim, LDSO and others have been used by many researchers for many ends. To name a few, Gao et al used GD techniques to Estimate of the Average Number of Recessive Lethal Mutations Carried by Humans (2015). Kennedy et al used GD to measure the rate of Inbreeding Depression among the Chatham Island Black Robin (2013).

In this project in particular we have followed Macluer et al's method very closely. We have included Figures 1 and 2 below from Macluer's paper. As per Macluer's method, GD begins with the assignment of two unique hypothetical alleles to each of the founding members of a colony. Figure 1 illustrates the process for a simple pedigree consisting only of animal A and his ancestors. A random number generator is used to determine which one of each founders genes is transmitted to each offspring. At the end of a single gene drop, every animal in the pedigree has a genotype (Fig. 2).

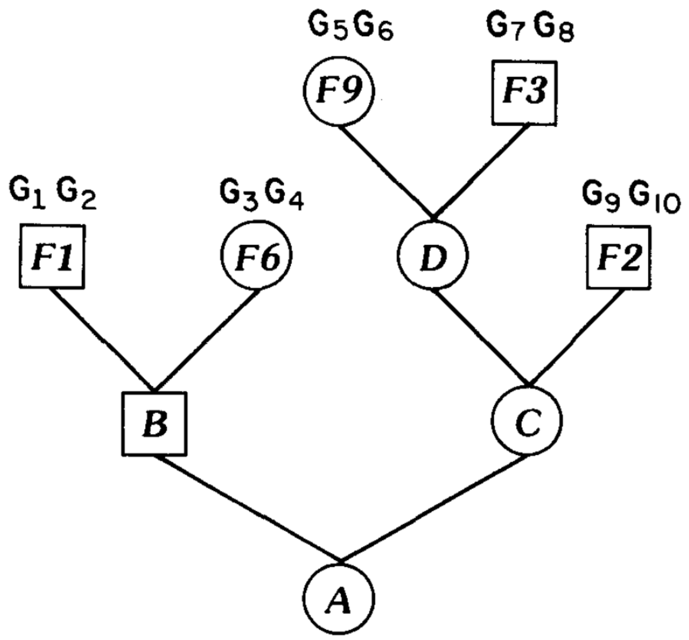


Fig. 1. Pedigree with two unique hypothetical alleles assigned to each founder.

Figure 1: Figure 1 from Macluer et al (1986).

Macluer's method performed thousands of drops at a time. The method in this project aims to be used on the order of several hundred thousand to a million drops, to generate richer data. The program developed through the course of this project is also highly customizable, as an improvement on Macluer's original implementation.

2 Work with Nancy Chen and the Clark Lab

The purpose for this project came about in the Summer of 2015. Nancy Chen, at the Clark Lab of Cornell University was conducting research that would be used to write a paper that tackles the question, "what is the importance of recent natural selection in governing patterns of genetic variation in a natural population?".

This project was seeking a customizable, fast implementation of Gene Dropping. The author and a fellow undergradaute studying Computer Science wrote a program in Python to simulate the GD Chen would need to answer this question. The developed program takes in a pedigree file, which encodes the necessary information as explained above. It also takes in a file mapping breeding pairs to the year in which they bred and a file mapping SNPs to the chromosomes they occur on (format explained in Appendix B). The program then "drops" genes down the pedigree, using Macluer's random approach. For each individual it randomly selects an allele at each SNP from each parent. Special cases are delineated further in the section on Methods. This program was written in Python.

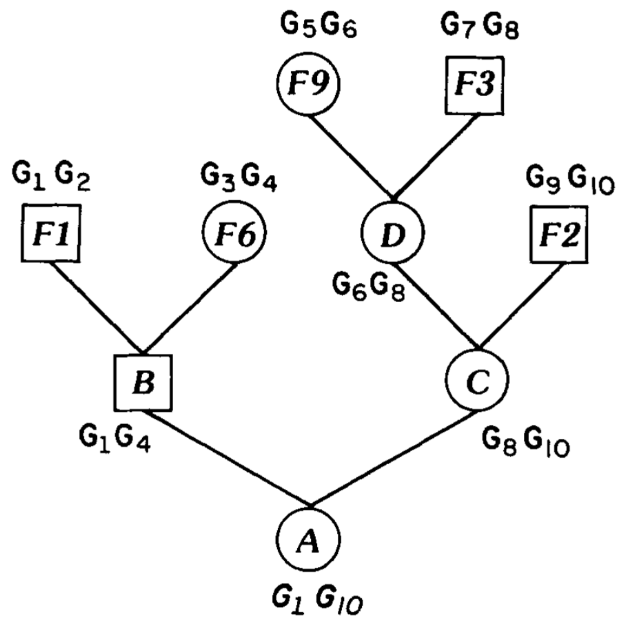


Fig. 2. Pedigree after gene dropping, with genotypes assigned to descendants by Mendelian segregation of founder alleles.

Figure 2: Figure 2 from Macluer et al (1986).

2.1 Issues and Shortcomings of this work

The program was written in Python for its usability and wealth of libraries for file manipulation and data handling. However Python's speed proved to be prohibitively expensive. The Python program could only perform at about 30 seconds a drop which amounts to almost 4 days to run 10,000 gene drops and more than a month to run 100,000 gene drops. Since this is the lower end of the magnitude of drops this program would be used for, this speed rendered the program useless for the purpose of this research.

Note: All drop time measurements were taken on a computer with 2.6 GHz processing power.

3 Extension: Speedup

Thus the primary purpose of this project is a speedup on the method developed by the author and others in the Clark Lab in 2015.

3.1 Method for achieving speedup

The Python implementation of the GD program is very robust in terms of preprocessing the data. Thus for this project we will use some of the preprocessing logic already written into the Python implementation. However preprocessing is only a one-time requirement, so it takes a constant time, after which, with clean enough data, a C program can perform the actual drops down the pedigree much faster than the Python dropping implementation.

Therefore the scope of this project was making use of the existing Python implementation to preprocess the data and writing a C implementation to perform the actual dropping logic. A necessary last step is displaying the results of the C implementation's drops, which is also in the scope of this project.

4 Method

This section will explain the entire GD implementation from preprocessing to displaying the data.

4.1 Preprocessing

The first step in Preprocessing is taking isolated individuals out of the pedigree. Isolated individuals are defined as individuals that appear in the pedigree that have no parents and no offsprings, either due to lack of data, or for any other reason.

The next step is dealing with founders. Founders are defined as the breeding pairs of individuals in the first breeding year under scrutiny. The first breeding year can either be input directly (eg: 2001), or calculated from the data. If the user selects the calculation from data option, the program chooses the first year as the year in which there is genotype data for at least a certain fraction (called the dense data threshold - also user specified) of breeding pairs in that year. The breeding pairs in the first year that meets this criteria are chosen as "founders". Founders are important because the founding generation's allele frequencies are treated as the baseline allele frequencies in data analysis. Founders' genotypes are the ones dropped throughout the pedigree, so these are the genotypes we care most about.

Therefore the program has settings built in to deal with missing founder genotype data. The user can select one of four options - impute, simulate, leave unknown, user specified. Impute means that if a founder's genotype data is missing, it is imputed from the genotypes of its known offsprings. Simulate means the founder is assigned likely genotypes based on the allele frequencies of other founders' genotypes. Leave unknown, is as the name suggests an option to leave unknown founders' genotypes unknown and drop an 'X' down the pedigree to denote the genes of this founder. User specified allows the user to provide one line of a pedigree file that has genotype data for all missing founders' genotype data.

The last step in preprocessing is dealing with immigrants. Immigrants are defined as individuals with offsprings that either have no parents, or have no parents that bred in the founding generation. These individuals are also interesting because they add genes to the population as the gene dropping continues, but they are not founders, so they might confuse the data analysis. In order to maintain the sanctity of the data analysis, the user also has the following options on how to deal with immigrants. The user can select one of five options - impute, simulate, leave unknown unknown, leave unknown all and user specified. Impute means that if an immigrants's genotype data is missing, it is imputed from the genotypes of its known offsprings. Simulate means the immigrant is assigned likely genotypes based on the allele frequencies of founders' genotypes. Leave unknown unknown, is an option to leave unknown immigrants' genotypes unknown and drop a 'U' down the pedigree to denote the genes of this immigrant. Leave unknown all, is like leave unknown unknown, but all the immigrants' genes (even known ones) are dropped as 'U' down the pedigree. User specified allows the user to provide one line of a pedigree file that has genotype data for all missing immigrants' genotype data.

4.2 Dropping and Data Collection

The drop step follows a simple inner-outer loop structure. The inner loop assigns alleles to each SNP of a particular individual based on a random coin flip from their mother's alleles at that SNP and the same for their father's. The outer loop runs through the entire pedigree, so that each individual gets a simulated genotype. It makes sure to simulate genotypes in an order such that when simulating an individuals' genotype, its parents always already have a simulated genotype.

As it assigns each individual a genotype, it also updates a data structure keeping track of the various counts of each allele at each SNP across all the breeders for each breeding year.

4.3 Post processing

The post processing consists of printing out the above data structure in the following format.

1. YEAR is the breeding year for which this line gives data. The data will be the information collected from all the breeders in that year.

2. SNP is the SNP id.

3. NUM_ALLELES is the number of observed alleles at this SNP. This will always be 1 or 2 unless the 'U' or 'X' options are acitvated.

4. ALLELE:COUNT... is each allele and its corresponding count.

5. TIMES_THESE_COUNTS_OCCURED is the number of times this configuration of counts occurred with this YEAR and SNP.

The user can also input a list of YEAR, SNP pairs or select 'all' if the user does this, in a separate file the p_beginning, and p_observed values for that YEAR, SNP pair are printed in a separate file. With these aspects of the data, the user can calculate p-values as in Figure 3 below. The user also has an option telling the program to generate graphs, in which case the graphs look like the ones shown in the section on Results.

5 Improvement on Macluer's work: Customizability

Clearly this program is highly customizable. The various aspects the user can control are as follows:

1. The pedigree to drop on.
2. The number of drops to perform.
3. Whether to calculate the founding year or specify it.
4. Whether to impute, simulate, leave unknown or specify missing founder data.
5. The same for immigrants.
6. Which, if any YEAR, SNP pairs to print more detailed data about.
7. Whether to output graphs or not.

6 Results

6.1 Results of attempted Speedup

The final speed (on a 2.6 GHz processor) is about 50 seconds for preprocessing, then about 0.2 seconds per drop and then about (0.05*number of drops) seconds to output the data correctly.

6.2 Example data generated

Figures 4 and 5 are snapshots of the sample output generated by running 1,000 gene drops on the entire pedigree. The entire file is also included with the submission as data/one_thousand.out.

The headings are explained in the subsection on Postprocessing in the section on Methods.

Figure 6 is an example of the p-value data outputted by the program.

1. YEAR is the year in question
2. SNP_NUM is the index of the SNP in question
3. SNP_ID is the id of the SNP in question
4. NUM_ALLELES_AT_SNP is the number of alleles seen at this SNP in any year.

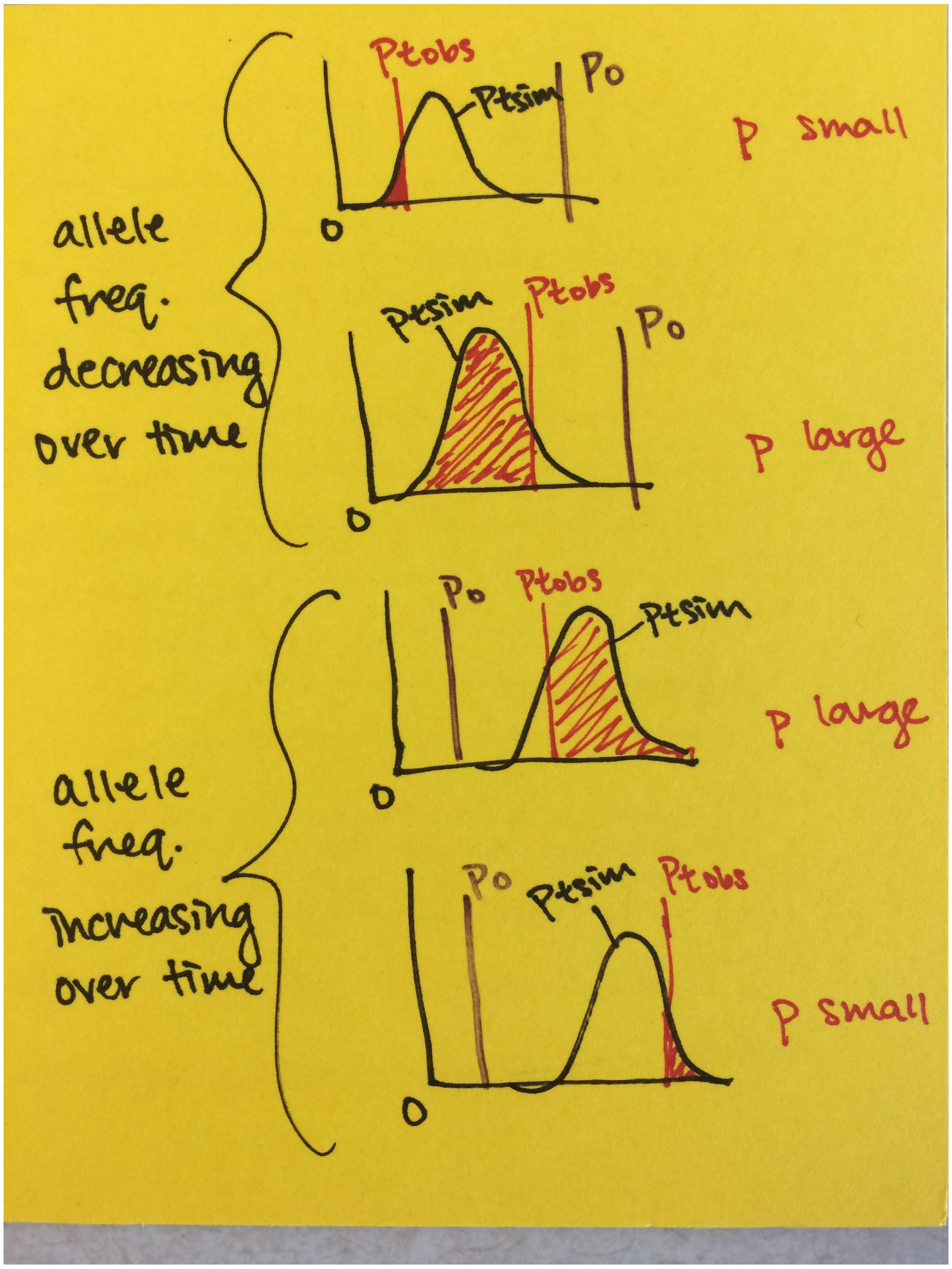
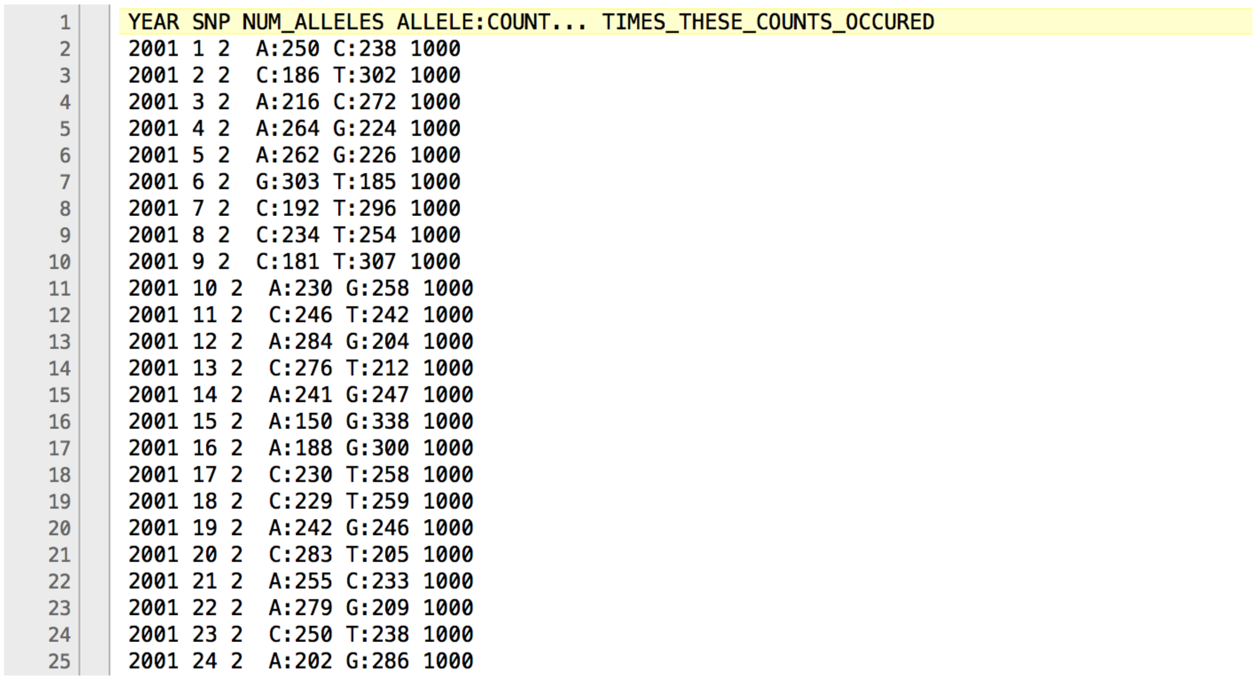


Figure 3: P-value calculation, courtesy of Nancy Chen.



```
YEAR SNP NUM_ALLELES ALLELE:COUNT... TIMES_THESE_COUNTS_OCCURED
2001 1 2  A:250 C:238 1000
2001 2 2  C:186 T:302 1000
2001 3 2  A:216 C:272 1000
2001 4 2  A:264 G:224 1000
2001 5 2  A:262 G:226 1000
2001 6 2  G:303 T:185 1000
2001 7 2  C:192 T:296 1000
2001 8 2  C:234 T:254 1000
2001 9 2  C:181 T:307 1000
2001 10 2  A:230 G:258 1000
2001 11 2  C:246 T:242 1000
2001 12 2  A:284 G:204 1000
2001 13 2  C:276 T:212 1000
2001 14 2  A:241 G:247 1000
2001 15 2  A:150 G:338 1000
2001 16 2  A:188 G:300 1000
2001 17 2  C:230 T:258 1000
2001 18 2  C:229 T:259 1000
2001 19 2  A:242 G:246 1000
2001 20 2  C:283 T:205 1000
2001 21 2  A:255 C:233 1000
2001 22 2  A:279 G:209 1000
2001 23 2  C:250 T:238 1000
2001 24 2  A:202 G:286 1000
```

Figure 4: Example output

5. ALLELES:P_FIRST_GENERATION... are these alleles with their frequencies in the founding generation.

6. ALLELES:P_OBSERVED... are these alleles with their frequencies in the observed generation.

Figure 7 is an example graph generated by the program.

7 Scientific question and other applications of this work

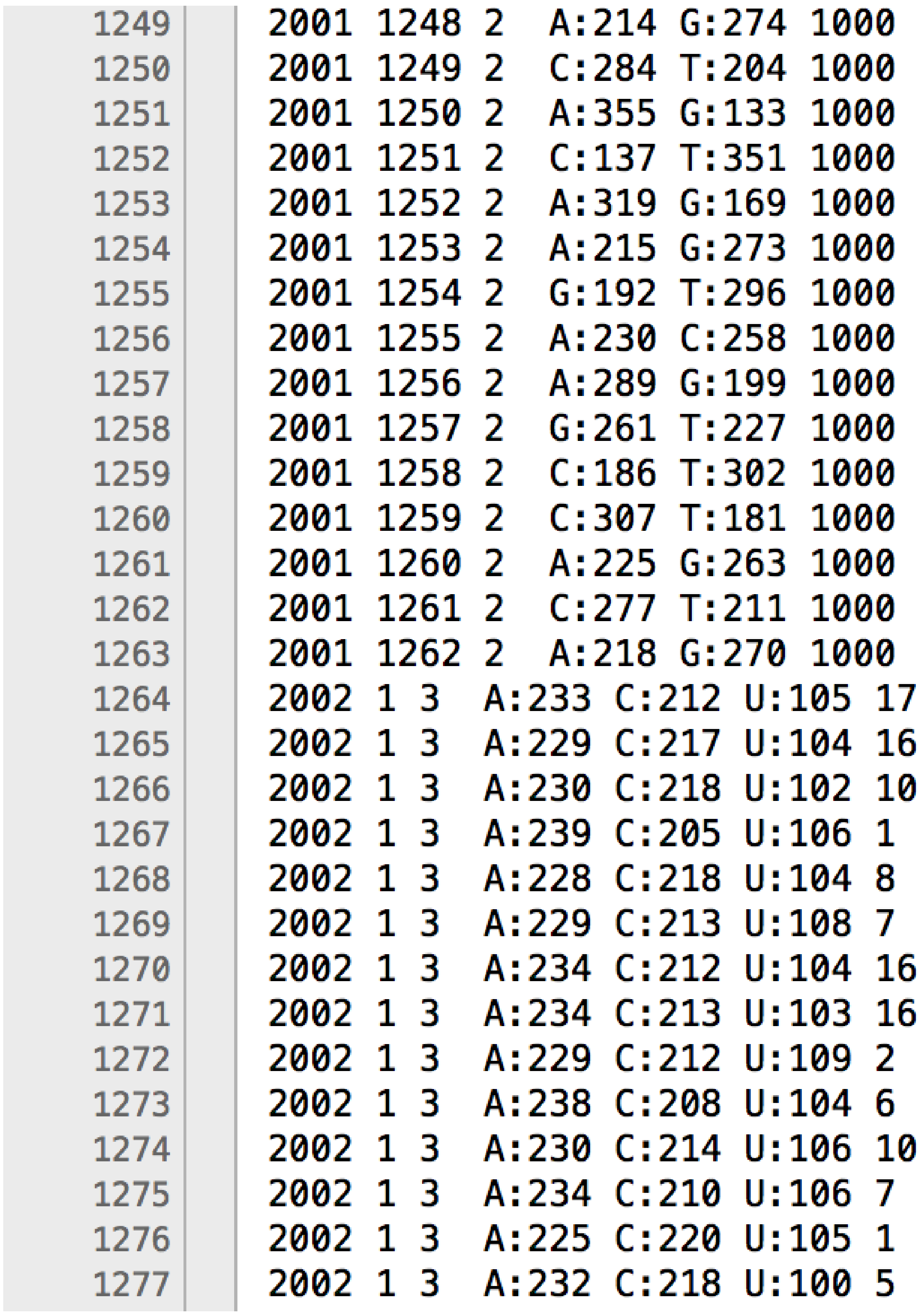
Clearly using this data technique one can find out which SNPs show anomalous behaviour, and make various inferences. Since we also have a map from SNPs to chromosomes we can make phenotypic inferences about anomalous genotypic behaviour at particular SNPs as shown by this data.

One can also correlate the phenotypes that correspond to changes in these SNPs and understand what traits might be selected for or evolutionarily beneficial.

This program will now be used by Chen et al to answer the original scientific question they set out to answer, ”what is the importance of recent natural selection in governing patterns of genetic variation in a natural population?”.

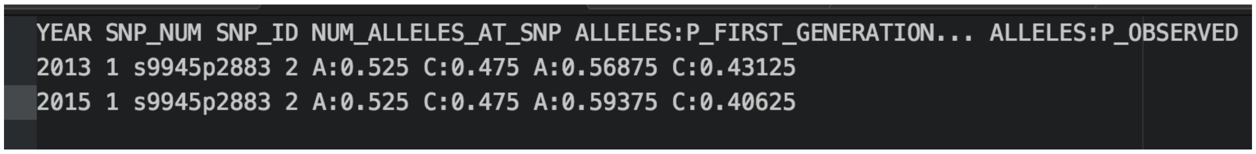
8 Extensions and Areas to improve the code

The section of the code to generate graphs and read user specified genotypes for founders or immigrants is a bit buggy as of now, and needs some attention. The graphs' headings and labeling



```
2001 1248 2  A:214 G:274 1000
2001 1249 2  C:284 T:204 1000
2001 1250 2  A:355 G:133 1000
2001 1251 2  C:137 T:351 1000
2001 1252 2  A:319 G:169 1000
2001 1253 2  A:215 G:273 1000
2001 1254 2  G:192 T:296 1000
2001 1255 2  A:230 C:258 1000
2001 1256 2  A:289 G:199 1000
2001 1257 2  G:261 T:227 1000
2001 1258 2  C:186 T:302 1000
2001 1259 2  C:307 T:181 1000
2001 1260 2  A:225 G:263 1000
2001 1261 2  C:277 T:211 1000
2001 1262 2  A:218 G:270 1000
2002 1 3  A:233 C:212 U:105 17
2002 1 3  A:229 C:217 U:104 16
2002 1 3  A:230 C:218 U:102 10
2002 1 3  A:239 C:205 U:106 1
2002 1 3  A:228 C:218 U:104 8
2002 1 3  A:229 C:213 U:108 7
2002 1 3  A:234 C:212 U:104 16
2002 1 3  A:234 C:213 U:103 16
2002 1 3  A:229 C:212 U:109 2
2002 1 3  A:238 C:208 U:104 6
2002 1 3  A:230 C:214 U:106 10
2002 1 3  A:234 C:210 U:106 7
2002 1 3  A:225 C:220 U:105 1
2002 1 3  A:232 C:218 U:100 5
```

Figure 5: Example output



```
YEAR SNP_NUM SNP_ID NUM_ALLELES_AT_SNP ALLELES:P_FIRST_GENERATION... ALLELES:P_OBSERVED
2013 1 s9945p2883 2 A:0.525 C:0.475 A:0.56875 C:0.43125
2015 1 s9945p2883 2 A:0.525 C:0.475 A:0.59375 C:0.40625
```

Figure 6: Example line of a p data output file

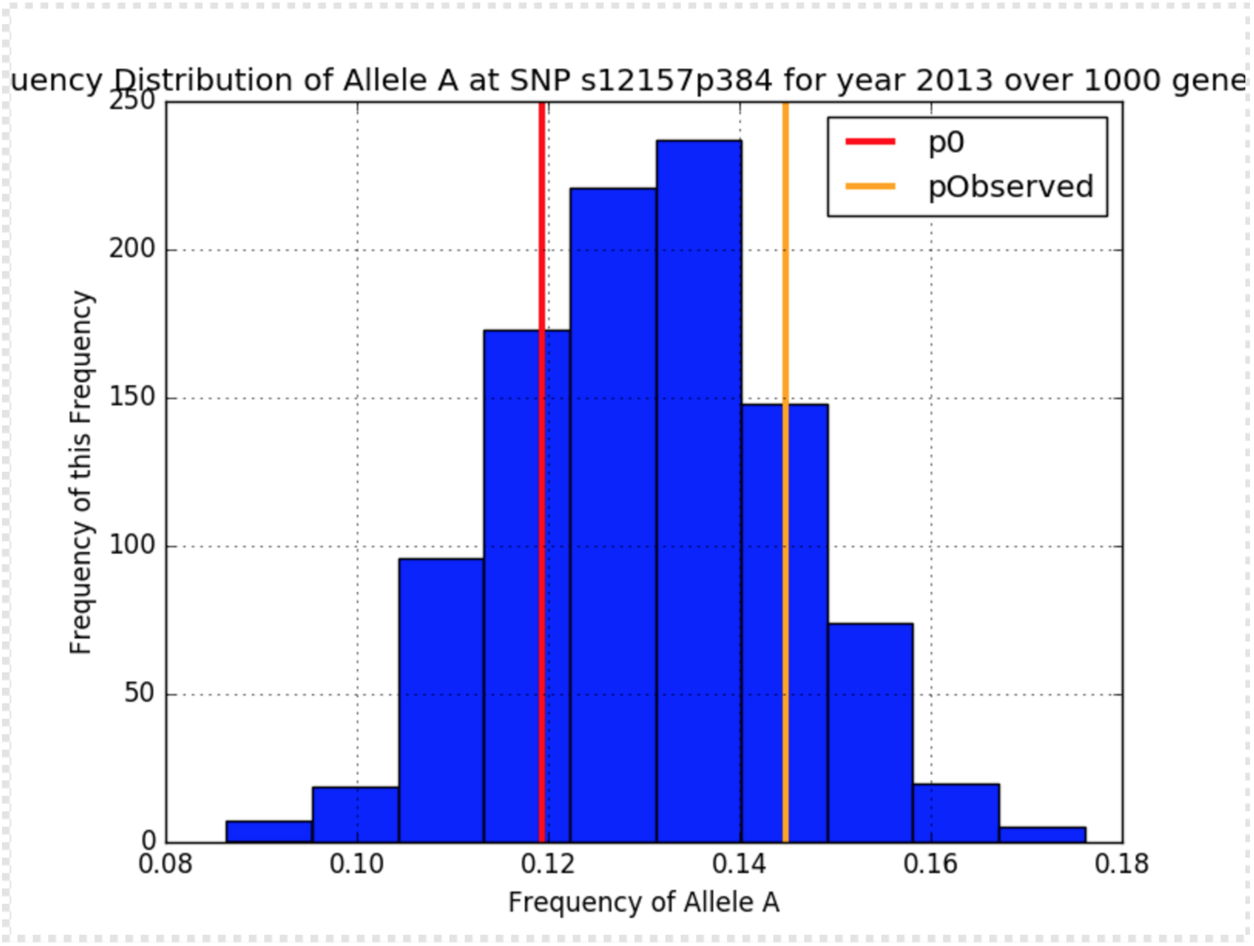


Figure 7: Example graph generated by prorgram from data output and p data output

can be improved too.

As of now the code treats autosomes and allosomes the same, but it can be extended to treat allosomes differently, since genes are not dropped similarly randomly at allosomes. The Python implementation dealt with this, so an extension to the C implementation would be adding functionality to deal with this.

Currently the post-processing, data output phase is not optimized at all. The data is output to a file, and then a Python script scans and cleans that file, collapsing identical lines, and incrementing their count. This step should be done programmatically for enhanced speed.

Another extension is outputting a metadata file, which has information about which and how many individuals were isolated, which and how many founders didn't have genotype data, which and how many immigrants there were etc.

Since we have SNP to chromose maps, two immediate extensions to this work are local phasing and haplotype inference. Using the observed data and seeing SNPs that behave similarly, one can look at the chromosomes they are on, and make inferences about recombination events.

9 References

1. *Pedigree Analysis by Computer Simulation.* MacCluer, VandeBerg, Read and Ryder. 1986. Research Department, San Diego Zoo, San Diego.

2. *An Estimate of the Average Number of Recessive Lethal Mutations Carried by Humans.* Gao, Waggoner, Stephens, Ober and Przeworski. 2015. Columbia University, New York, New York.

3. *Severe inbreeding depression and no evidence of purging in an extremely inbred wild speciesthe chatham island black robin.* Kennedy, Grueber, Duncan and Jamieson. 2013. Institute for Applied Ecology, University of Canberra, Australia.

4. **https://en.wikipedia.org/wiki/Mendelian_inheritance**. For information on Mendel's laws.

I would like to thank Nancy Chen and Andrew Clark for introducing me to this project. I would like to thank Ashish Akshat for co-writing the first attempt at the project with me.

10 Appendices

10.1 Appendix A: Configuring and running the code

The code is included as geneDropping.zip. Unizipping this directory and navigate to it in a Terminal session.

1. type cd src. Hit enter.

2. Edit the following files:

   2.1. constants.py:

2.1.1. DENSE_DNA_THRESHOLD: Edit to desired value. This is the fraction of breeders to have data for a year to be considered the founding year.

2.1.2. FOUDING_YEAR: Can specify user defined founding year or leave as 0.

2.1.3. IMPUTE_IMM: True or False depending on whether you want to impute immigrant alleles or not.

2.1.4. IMM: 'sim', 'ua', 'uu' or filename for the simulate, unknown unknown, unknown all and specify options explained above.

2.1.5. IMPUTE_FOUNDERS: True or False depending on whether you want to impute founder alleles or not.

2.1.6. FOUNDER: 'sim', 'x' or filename for the simulate, unknown and specify options explained above.

2.1.7. P_VALUES_FOR: list of YEAR SNP pairs. SNPs can be specified either as indices or as string SNP ids. Can be blank.

2.1.8. P_GRAPH: True if the program should output a graph, False otherwise. Invalid if P_VALUES_FOR list is empty.

2.1.9. PEDFILE: The pedigree file.

2.1.10. MAPFILE: The file mapping SNPs to Chromosomes.

2.1.11. BREEDERFILE: The file mapping years to the pairs of individuals that were breeders in that year.

2.2. geneDrop.h:

2.2.1. gene_length: the number of SNPs.

2.2.2. founding_year: the founding year, if user specified. Leave 0 otherwise.

2.2.3. N: the number of drops to simulate.

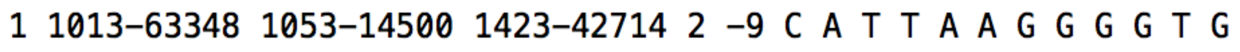
3. type chmod 755 runGeneDropping. Hit Enter.

4. type./runGeneDropping. Hit Enter. Note this will print the data onto the terminal, to pipe the output to a file type ./runGeneDropping ¿ filename.ext.

10.2 Appendix B: Input File formats

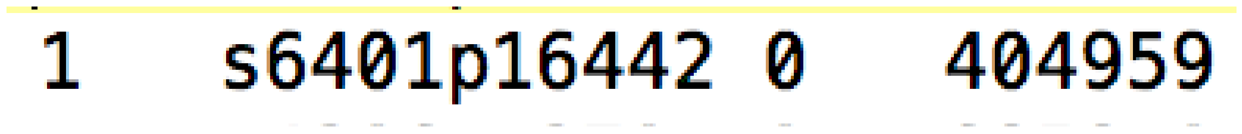
PED FILE: Figure 8.

1. ”1” is the family ID.
2. ”1013-63348” is the individual ID.
3. ”1053-14500” is the father's ID in the ZW system and the mother's in the XY system.
4. ”1423-42714” is the mother's ID in the ZW system and the father's in the XY system.
5. ”2” is the sex, in this case Male.
6. ”-9” is phenotypic information.
7. ”C A ...” is the genotype



```
1 1013-63348 1053-14500 1423-42714 2 -9 C A T T A A G G G G T G
```

Figure 8: Example line of a .ped file



```
1    s6401p16442 0    404959
```

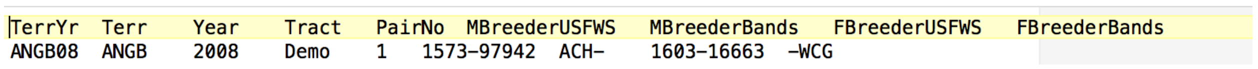
Figure 9: Example line of a .map file

MAP FILE: Figure 9

1. ”1” is the chromosome number.
2. ”s6401p16442” is the SNP id.

The other two lines are ignored for the purposes of this project.

BREEDER FILE: Figure 10. We are interested in ”Year”, ”MBreederUSFWS” and ”FBreederUSFWS”



```
TerrYr  Terr    Year    Tract   PairNo  MBreederUSFWS   MBreederBands   FBreederUSFWS   FBreederBands
ANGB08  ANGB    2008    Demo    1   1573-97942  ACH-    1603-16663  -WCG
```

Figure 10: Example line of the breeder mapping file