Haplotype Phasing Algorithm

Data Imputation - Michael Cheng

The data imputation method used is a K Nearest Neighbors imputation method provided by the "fancyimpute" python package. The function knn_imputer imputes missing data based on the nearest rows of SNPs that have similar values with the row currently being imputed. The element with the missing value by finding the most common value at that element of the similar rows of SNPs. After the genotype data was read in as a pandas data frame, the * symbol in the genotype data were first converted into "nan" values. Since the KNN function acts on quantitative data, the genotype data was read numerically, and the average of the nearest neighbors' elements was rounded and imputed. The imputed dataset was then outputted into the text file to be used to construct the potential haplotype phase matrix.

Data Input - Anweshan Das

The full imputed data was read from the text file using the standard readlines command, which input the genotypes data into a list of strings, one string for each SNP.

Constructing The Potential Haplotype Phases Matrix - Anweshan Das

Before we could run Clark's algorithm, we had to convert the data from the input format, as detailed in the Data Input section, into a format more apt for running Clark's algorithm on. This format was to be a 4-dimensional tensor people, which will be explained later. First, we had to divide the 39267 SNP's into 4-5 SNP sections, with the first 39255 being 5 SNP sections and the last 12 being 3 4 SNP sections, each section with 50 people. The string format of these sections were converted into a two-dimensional matrix: lists of people, each person with a list of 4-5 numbers for the next 4-5 SNPs. To first generate the possible phases for each 4-5 SNP section, we counted the number of ones present in a 4-5 SNP sequence for each person. 2^(number of ones), or 2^((number of ones)-1) haplotype pairs of, haplotype phases were possible per sequence per person. To generate the 4-dimensional haplotype phase matrix, we took each person (first dimension was a list of people), took each 4-5 SNP sequence per person (second dimension was a list of haplotype pairs), generated pairs of haplotypes (the third dimension, each with a list of a pair of haplotypes, or a sole haplotype is there were 0 1's in the genotype sequence), and the 4th dimension was a list of 4-5 numbers representing the possible presences for each of the 4-5 SNP's.

Clark's Algorithm - Michael Cheng

Clark's Algorithm is the algorithm we used to predict the true haplotype of the genotype data. The parameters input into the function call are the potential haplotype phases matrix, which gives the potential phases of genotypes for all 50 people, and a haplotype list to append the predicted haplotypes. At the start of the function, an empty known list of haplotypes is created. The potential haplotype phase list is iterated through by column, or people. If there is only one

potential pair of haplotypes to choose from, then that pair is appended to the final haplotype list and also written into the known list of haplotypes. Otherwise, each haplotype pair is compared to the known haplotypes and assigned a number corresponding to the number of matches the haplotype pair has with the known list (either 0, 1, or 2). After all pairs are evaluated for the person, the pair with the maximum number of matches is chosen. If there are multiple pairs with the same maximum, the first pair encountered will be chosen. Any haplotype of the pair is added to the known list of haplotypes if not already in the list. The algorithm is performed every time 4-5 SNPs have been evaluated for its phases for all 50 people. This means that Clark's Algorithm is performed after each iteration of 50 people during the construction of the potential haplotype phases matrix. The final output list will be converted to the correct output format after the full potential haplotype phasing matrix is created.

Constructing The Output - Anweshan Das

The output of the Clark's algorithm function was a 785,400 long list of SNP columns, each a list of numbers, or 7854 (the number of SNP's divided by 100, for 100 haplotypes, divided by 5, the number of SNP's per segment) segments of 100 haplotypes. This segment took the first fifty of each 100 haplotype segment and combined them with the second fifty of each haplotype segment for implementation purposes of immutable python strings; we got around the immutability and output string formatting difficulties using a cipher. For our purposes, this implementation detail is not important. However, it is important that the goal of the algorithm is to first iterate through the top (of the 5 rows) of the first 100 haplotypes, then the 2nd row of the first 100 haplotypes…until the 5th row of the first 100 haplotypes. This generates the first 5 lines/SNPS, or strings, of text output. Then we iterate for the next 100 to generate the next 5 lines/SNPS of text output, and so on. It is of minor detail that because the number of SNP's was not divisible by 5, that the last 300 haplotypes were lists of 4 SNPS, which was an additional few slightly different iterations. The formulas and floor function were technicalities of the matrix transformation, and the replace function replaces the given cipher position in a premade string with the number expected to be at that position by Clark's algorithm.

Data Ouput - Anweshan Das

The data was outputted into the output file as a list of strings, each string with a newline at the end, using the standard writelines command.

Packages Used:

Fancyimpute: https://github.com/iskandr/fancyimpute
Pandas: https://pandas.pydata.org/
Numpy: https://numpy.org/