Class project

- Due 12/18/2020 (Friday) 5 pm in LA time.
- There are five projects and choose one of them.
- You may use any programming language.
- You need to submit a 2-3 page long report describing the problem, the approach you took to solve the problem, the results you got, and issues you had and potential improvements.
- You also need to submit code/scripts and instruction on how to run your code.
- The format of report is not defined, but you may have following sections (like a research paper).
  - Introduction: describe background information about the problem you are trying to solve.
  - Method: describe the approach you took to solve the problem. Do not include code/scripts in this section (or anywhere in the report). Your code/script does not count as pages for your report.
  - Results: show the results you got. You need to have at least two figures/tables (could be one figure and one table or two figures or two tables) and describe your results in words. Do not make your figures/tables too large or small.
  - Discussion: summarize your method and results. Describe any potential improvements or any issues you encountered.
- Project details are subject to change.

Class project

- Project grading will be based on the following*:

1. Completeness of the project (e.g. Were you able to do all the analyses in the project?).
2. Quality of a report (e.g. Is the report clearly and well written? Do tables/figures show results clearly and accurately?).
3. Approaches (e.g. is the algorithm and/or its implementation correct?).
4. Results (e.g. are the results you got correct?)
5. Designing experiments: For some projects, you need to choose certain parameters in the experiments or design your own experiments. This grading criterion will be based on how well you design the experiments (or choose the parameters).

*Subject to change.

Project #1. Genome assembly using the de Brujin graph.

In this project, you will generate a random human genome and implement the de Brujin graph algorithm we learned in class to assemble the genome. First, generate a human genome with 1,000 nucleotides with random As, Cs, Ts, and Gs (random ordering). Next, generate k-mers from this genome using read length of 10 (let X be the read length). Using those reads, construct the de Brujin graph using those k-mers and assemble the genome. Can you construct the same genome? What happens if you change X?

Next, add repeats to the genome. Repeats can be length of 20 (let Y be the length of repeat), and it appears at least 10 times (let Z be the number of repeats). How does the repeat influence your assembly? Can you construct the same genome? If not, can you do the random walk in the de Brujin graph and come up with the path that leads to the original sequence? How many random walks do you need to do? You may need to repeat this analysis several times to find how many random walks you need on average to find the original sequence. Vary X, Y, and Z and see how different values of X, Y, and Z influence your results (the average random walks). You do not need to assume that X is always smaller than Y or always greater than Y.

Project #1. Genome assembly using the de Brujin graph.

(Graduate students): Please choose one of the following ideas.

Idea 1. Calculate the number of all Eulerian cycles using the BEST theorem (pg 181-182 in textbook). How does the number of Eulerian cycles change for different values of X, Y, and Z?

Idea 2. Add random mutations to k-mers. How does this influence the assembly (with and without repeats)?

Idea 3. Increase genome size to 1 million nucleotides.

Project #2. Sequence alignment using the dynamic programming.

In this project, you will implement dynamic programming to compare two sequences. You are given a following scoring matrix that specifies match, mismatch, and indel scores.

	A	C	G	T	–
A	+matchAA	–mismatchAC	–mismatchAG	–mismatchAT	–indel
C	–mismatchAC	+matchCC	–mismatchCG	–mismatchCT	–indel
G	–mismatchAG	–mismatchCG	+matchGG	–mismatchGT	–indel
T	–mismatchAT	–mismatchCT	–mismatchGT	+matchTT	–indel
–	–indel	–indel	–indel	–indel	

Given this matrix, implement both global and local alignments. Also, implement each alignment with and without the middle node/edge algorithm. Show your implementation (both with and without middle node/edge algorithm) generates correct alignment using some toy examples. Also, your code should generate all possible alignments with the same score if there is more than one alignment.

Project #2. Sequence alignment using the dynamic programming.

Show the performance of your alignment algorithm in terms of runtime and memory usage as the string size increases. What is the maximum length of strings you can compare (sum of lengths of two strings)? You need to show the performance (runtime and memory usage) when using and when not using the middle node/edge algorithm. You need to run the alignment multiple times across different sets of strings and calculate average runtime and/or memory usage. How much does the middle node/edge algorithm improve the performance for different lengths of sequences?

(Graduate students): Please choose one of the following ideas.

Idea 1. Additionally, implement multiple alignment problem using the greedy algorithm. Describe your algorithm and how it performs.

Project #2. Sequence alignment using the dynamic programming.

Idea 2. The middle node algorithm allows us to split the problem space initially into 2 or subproblems that can be computed independently and then added together. This allows us to split the work into 2 concurrent units that cuts the time down in half. By why stop at 2? Develop a new algorithm based off of the middle node algorithm that utilizes multiple cpu threads, concurrent processes, MapReduce nodes, and/or gpu threads to split the amount of work needed to compute the global alignment using the middle node algorithm. Show the real word performance gains as a function of concurrent units and input size.

Idea 3. Suppose we change the indel penalty to be a function of the score provided by the matrix above and the length of number of previous consecutive indels. That is, the alignment will take as input a function f(n), with n being the length of previous indels, and the matrix above. Develop a new sequence alignment algorithm to compute the optimal global alignment based off of the above-mentioned inputs.

Project #3. Genome-wide association studies (GWAS).

In this project, you will perform GWAS on a certain phenotype and find SNPs that are associated with the phenotype. The data you will work with is called 1000 genomes project; unlike its name suggests, there are actually 2,504 individuals from many different countries in the world. It is a low-coverage whole-genome sequencing data, and I removed rare SNPs (whose MAF is < 15%) and removed SNPs that are highly correlated. In the end, the SNP data (or genotype data) that you will work include 2,504 individuals and 828,325 SNPs. The SNP data is in the binary PLINK format, and you can find more information on this format here (http://zzz.bwh.harvard.edu/plink/data.shtml). I simulated phenotype data for the 2,504 individuals. You can think of it as HDL cholesterol levels of individuals although data are transformed (in other words, it is not typical HDL cholesterol level).

For this project, you need to perform the following analyses.

Project #3. Genome-wide association studies (GWAS).

Input files posted to CCLE:

1. snpdata.bed: Binary PLINK file that contains genotype information for all individuals.
2. snpdata.bim : Text file that contains SNP information.
3. snpdata.fam : Text file that contains individual information.
4. phenotype.txt : Text file that contains phenotype information.

Project #3. Genome-wide association studies (GWAS).

- Each row in fam file is information on each individual.
- There are 6 columns in each row.

1. Family ID (same as individual ID in snpdata.fam).
2. Individual ID.
3. Father ID (0 because all individuals are unrelated in snpdata.fam).
4. Mother ID (0 because all individuals are unrelated in snpdata.fam).
5. Sex (1 for male and 2 for female).
6. Phenotype (Do not use this column for association! Use provided phenotype file).

```
HG00096 HG00096 0 0 2 1
HG00097 HG00097 0 0 2 1
HG00099 HG00099 0 0 2 1
HG00100 HG00100 0 0 2 1
HG00101 HG00101 0 0 2 1
HG00102 HG00102 0 0 2 1
HG00103 HG00103 0 0 2 1
HG00105 HG00105 0 0 2 1
HG00106 HG00106 0 0 2 1
HG00107 HG00107 0 0 2 1
```

Project #3. Genome-wide association studies (GWAS).

- Each row in bim file is information on each SNP.
- There are 6 columns in each row.

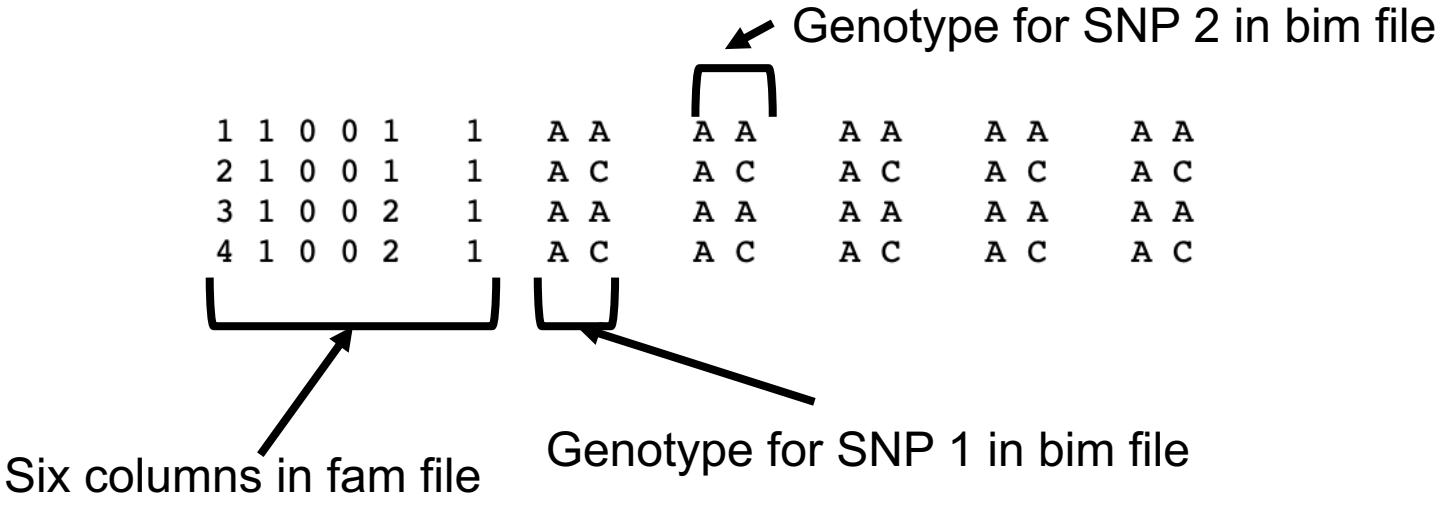
1. Chromosome.
2. SNP ID (also called RSID).
3. Genetic distance in morgan (0 because we do not use this information).
4. Base-pair position in bp units (position of SNP).
5. One allele (not necessarily minor or reference allele).
6. The other allele (not necessarily minor or reference allele).

- In PLINK file, there is no concept of reference allele!

```
1       rs2977608       0       768253  C       A
1       rs142008205     0       771410  C       T
1       rs56327836      0       787185  G       A
1       rs10900604      0       798400  G       A
1       rs7553096       0       802026  A       G
1       rs11240779      0       808631  A       G
1       rs4475692       0       825069  G       C
1       rs13303179      0       825410  G       A
```

Project #3. Genome-wide association studies (GWAS).

- Bed file is a binary file, so you cannot open it with text editor.
- PLINK also has .ped file, which is in a text format. Bed format is basically binary version of ped file without the first six columns (which are stored in fam file).
- Ped file has following information.



Project #3. Genome-wide association studies (GWAS).

- Each row in phenotype is phenotype information for each individual.
- There are 3 columns in each row.

1. Family ID.
2. Individual ID.
3. Phenotype value.

- This format is how PLINK encodes phenotype information.

```
HG00096 HG00096 0.205794
HG00097 HG00097 0.194969
HG00099 HG00099 0.112899
HG00100 HG00100 0.528705
HG00101 HG00101 -0.0304550
HG00102 HG00102 0.484799
HG00103 HG00103 0.144476
HG00105 HG00105 0.0447560
HG00106 HG00106 0.359374
HG00107 HG00107 -0.0514700
```

Project #3. Genome-wide association studies (GWAS).

1) Using SNP data and phenotype data, perform the linear regression using --linear option in PLINK.

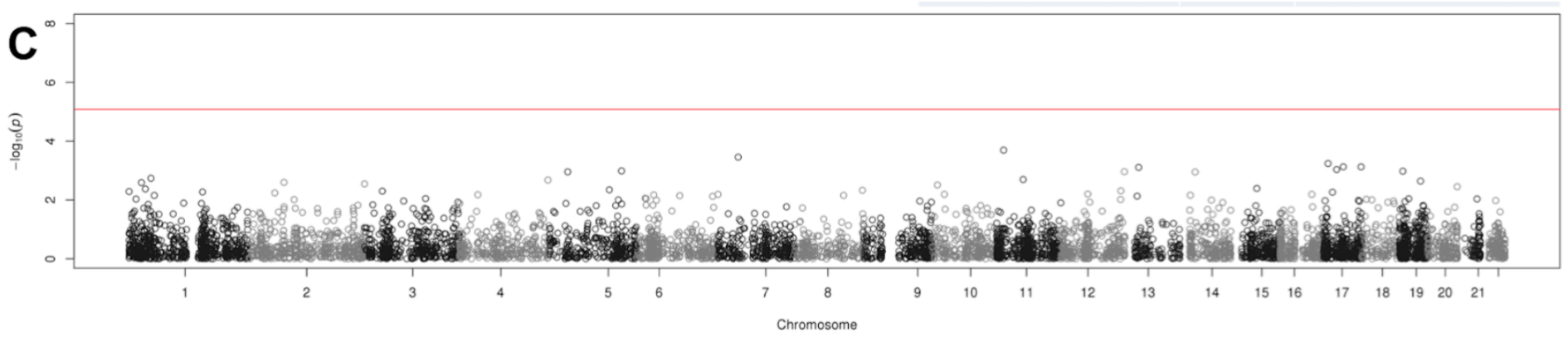
(You need to use --pheno option to specify phenotype file)

2) Convert the PLINK binary format to text format (using --recode option). Then implement linear regression in R and apply it to the SNP data (converted from PLINK binary format) and phenotype data. Find whether p-values from linear regression in R are consistent with p-values from linear regression in PLINK. You may want to calculate the differences in p-values between R and PLINK and plot the differences. If they are different, explain why they are different.

Project #3. Genome-wide association studies (GWAS).

3) Draw the Manhattan plot using p-values from linear regression results from PLINK. Use the provided R script (qqman.r) for this.

Example of Manhattan plot from other project



Project #3. Genome-wide association studies (GWAS).

4) Find a SNP with the minimum p-value in each chromosome from linear regression results from PLINK. Compare this p-value to p-value from UK Biobank GWAS summary statistics, which is available(https://docs.google.com/spreadsheets/d/1kvPoupSzsSFBNSztMzl04xMoSC3Kcx3CrjVf4yBmESU/edit?ts=5b5f17db#gid=178908679). Look for “HDL cholesterol (quantile)” and both_sexes results. Does the SNP with the minimum p-value in each chromosome from your GWAS have similarly low p-values in UK Biobank GWAS summary statistics? Create the table that lists the SNP with minimum p-value on each chromosome from your GWAS, the p-value from your GWAS and the p-value from UK Biobank GWAS summary statistics.

UK Biobank GWAS: Large-scale GWAS using UK Biobank data over 360,000 individuals. For more information, https://www.ukbiobank.ac.uk and http://www.nealelab.is/uk-biobank/faq

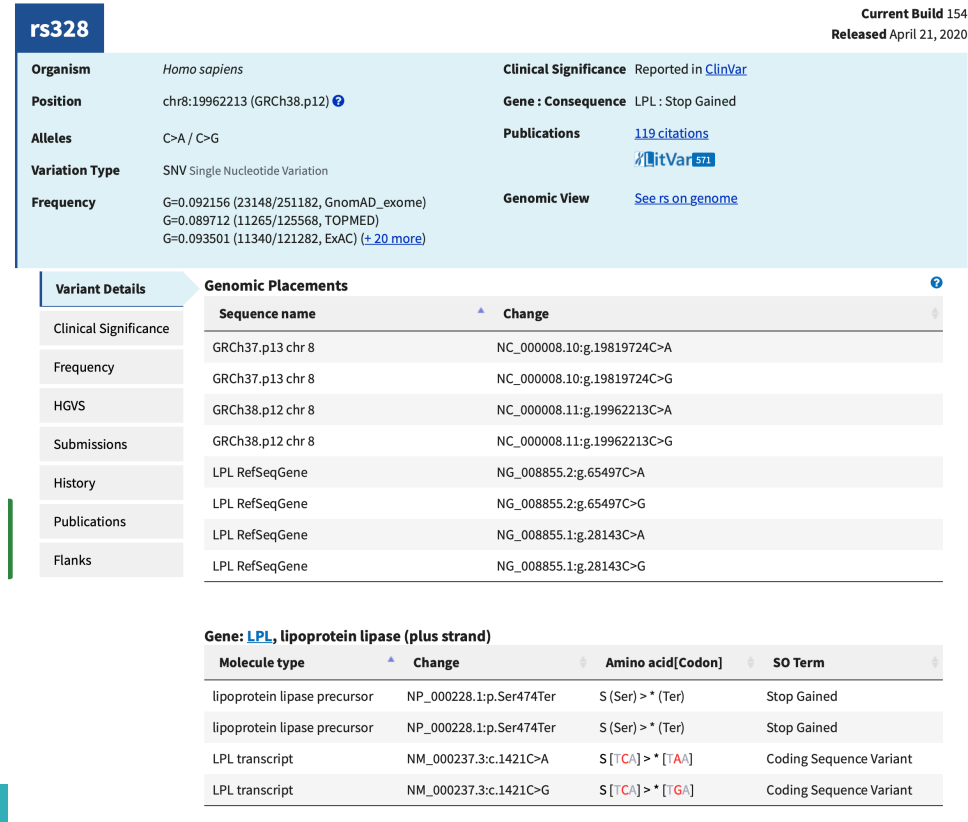
variant	minor_allele	minor_AF	low_confidence_variant	n_complete_samples	AC	ytx	beta	se	tstat	pval
10:10000018:A:G	G	4.4770e-01	false	315133	2.8217e+05	-4.0430e+02	-1.3146e-03	2.2927e-03	-5.7337e-01	5.6639e-01
10:100000625:A:G	G	4.3275e-01	false	315133	2.7275e+05	6.9828e+02	3.4570e-03	2.2921e-03	1.5082e+00	1.3150e-01
10:100000645:A:C	C	2.0558e-01	false	315133	1.2957e+05	-2.9326e+02	3.5318e-04	2.8141e-03	1.2551e-01	9.0012e-01
10:100003785:T:C	C	3.6077e-01	false	315133	2.2738e+05	-4.1663e+02	-4.0744e-03	2.3684e-03	-1.7203e+00	8.5378e-02
10:100004360:G:A	A	2.0558e-01	false	315133	1.2957e+05	-2.9582e+02	3.2554e-04	2.8141e-03	1.1568e-01	9.0790e-01
10:100004441:G:C	G	3.6154e-01	false	315133	4.0240e+05	4.0457e+02	3.9442e-03	2.3674e-03	1.6660e+00	9.5705e-02
10:100004906:C:A	A	4.3273e-01	false	315133	2.7274e+05	6.9449e+02	3.4371e-03	2.2920e-03	1.4996e+00	1.3373e-01
10:100004996:G:A	A	3.6074e-01	false	315133	2.2736e+05	-4.1757e+02	-4.0842e-03	2.3684e-03	-1.7244e+00	8.4629e-02

Project #3. Genome-wide association studies (GWAS).

5) Find if any SNP with the minimum p-value in the above table is a known SNP for HDL (have previous studies found the same SNP associated with HDL before?). You may want to use dbSNP database for this analysis.

dbSNP: database for SNPs at

https://www.ncbi.nlm.nih.gov/snp/



rs328

Current Build 154
Released April 21, 2020

Organism	*Homo sapiens*	**Clinical Significance**	Reported in ClinVar
Position	chr8:19962213 (GRCh38.p12)	**Gene : Consequence**	LPL : Stop Gained
Alleles	C>A / C>G	**Publications**	119 citations LitVar 571
Variation Type	SNV Single Nucleotide Variation		
Frequency	G=0.092156 (23148/251182, GnomAD_exome) G=0.089712 (11265/125568, TOPMED) G=0.093501 (11340/121282, ExAC) (+ 20 more)	**Genomic View**	See rs on genome

- Variant Details
- Clinical Significance
- Frequency
- HGVS
- Submissions
- History
- Publications
- Flanks

Genomic Placements

Sequence name	Change
GRCh37.p13 chr 8	NC_000008.10:g.19819724C>A
GRCh37.p13 chr 8	NC_000008.10:g.19819724C>G
GRCh38.p12 chr 8	NC_000008.11:g.19962213C>A
GRCh38.p12 chr 8	NC_000008.11:g.19962213C>G
LPL RefSeqGene	NG_008855.2:g.65497C>A
LPL RefSeqGene	NG_008855.2:g.65497C>G
LPL RefSeqGene	NG_008855.1:g.28143C>A
LPL RefSeqGene	NG_008855.1:g.28143C>G

Gene: LPL, lipoprotein lipase (plus strand)

Molecule type	Change	Amino acid[Codon]	SO Term
lipoprotein lipase precursor	NP_000228.1:p.Ser474Ter	S (Ser) > * (Ter)	Stop Gained
lipoprotein lipase precursor	NP_000228.1:p.Ser474Ter	S (Ser) > * (Ter)	Stop Gained
LPL transcript	NM_000237.3:c.1421C>A	S [TCA] > * [TAA]	Coding Sequence Variant
LPL transcript	NM_000237.3:c.1421C>G	S [TCA] > * [TGA]	Coding Sequence Variant

Project #3. Genome-wide association studies (GWAS).

(Graduate students): Please choose one of the following ideas.

Idea 1. Perform principal component analysis (PCA) using EIGENSTRAT software on the SNP data and draw the PCA plots showing the population structure of individuals from 1,000 genome data. For more information on PCA plots, read these papers (https://portlandpress.com/emergtoplifesci/article-abstract/3/4/399/219712/Variant-calling-and-quality-control-of-large-scale and https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2735096/).

Idea 2. Using UK Biobank GWAS summary statistics and provided 1000 genomes SNP data, calculate polygenic risk score (PRS) of individuals from the 1,000 genome data with PRSice-2 software (https://www.prsice.info). In PRSice-2, “Base dataset” refers to UK Biobank GWAS summary statistics and “Target dataset” refers to the 1000 genomes SNP data. Calculate R2 (variance explained by PRS) and its p-value for several p-value thresholds (PT) such as 5e-8, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-3, and 1e-2. How does R2 change as you increase the p-value thresholds?

Project #3. Genome-wide association studies (GWAS).

(Graduate students): Please choose one of the following ideas.

Idea 3. The SNP data include all individuals from different countries, and this may cause population structure, which may cause spurious associations between SNPs and phenotype. One approach to correct for population structure is using linear mixed models (LMMs). Apply EMMAX (https://genome.sph.umich.edu/wiki/EMMAX) to calculate p-values using LMMs and compare p-values from EMMAX and p-values from linear regression in PLINK. How do p-values change? Draw the Manhattan plot of p-values from EMMAX.

Project #4. Combinatorial pattern matching.

In this project, you will implement pattern matching algorithms to map reads to the reference genome. First, generate a human genome with 1 million nucleotides with random As, Cs, Ts, and Gs (random ordering). Then, generate 100,000 reads (let this be X) with length of 100 nucleotides (let this be Y) from this genome. Implement both PatternMatchingWithSuffixArray algorithm in page 484 of the textbook and BetterBWMatching algorithm in page 505 of the textbook. Apply both algorithms to the reads and genome you generated and compare the runtime and memory usage between suffix array and BWT for different X and Y.

Project #4. Combinatorial pattern matching.

(Graduate students): Please choose one of the following ideas.

Idea 1. BetterBWMatching algorithm does not allow us to find where the reads are located in the genome. Implement the partial suffix array with different values of K described in page 506 of the textbook and measure the performance (runtime and memory usage) for different values of K and also compare the performance to that of PatternMatchingWithSuffixArray. Implement also checkpoint arrays with different values of C to further improve the memory usage described in page 507 of the textbook and measure the performance (runtime and memory usage) for different values of C. What is the best value of K and C?

Idea 2. Solve the multiple approximate pattern matching problem using the algorithm described in pages 508 – 510 of the textbook. How does the performance change for different values of d (# of mismatches)?

Project #5. Efficient storage of genetic variant data.

- Before discussing this project, let's discuss Variant Call Format (VCF).
- After processing next generation sequencing data, genetic variants are stored in VCF.
- VCF is somewhat similar to PLINK, but VCF allows more information to be stored.
- VCF starts with headers with #.
- Headers can be arbitrary length and it has some meta-information.
- After headers, each line is each SNP (or genetic variants).
- The first 9 columns of each row has information on each SNP.
- 10^{th} column indicates variant information for individual 1, 11^{th} column indicates variant information for individual 2, and so on

Project #5. Efficient storage of genetic variant data.

header

1.1 An example

```
##fileformat=VCFv4.2
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
```

Individual 1

#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	NA00001	NA00002	NA00003
20	14370	rs6054257	G	A	29	PASS	NS=3;DP=14;AF=0.5;DB;H2	GT:GQ:DP:HQ	0|0:48:1:51,51	1|0:48:8:51,51	1/1:43:5:.,
20	17330	.	T	A	3	q10	NS=3;DP=11;AF=0.017	GT:GQ:DP:HQ	0|0:49:3:58,50	0|1:3:5:65,3	0/0:41:3
20	1110696	rs6040355	A	G,T	67	PASS	NS=2;DP=10;AF=0.333,0.667;AA=T;DB	GT:GQ:DP:HQ	1|2:21:6:23,27	2|1:2:0:18,2	2/2:35:4
20	1230237	.	T	.	47	PASS	NS=3;DP=13;AA=T	GT:GQ:DP:HQ	0|0:54:7:56,60	0|0:48:4:51,51	0/0:61:2
20	1234567	microsat1	GTC	G,GTCT	50	PASS	NS=3;DP=9;AA=G	GT:GQ:DP	0/1:35:4	0/2:17:2	1/1:40:3

Information on each SNP

Project #5. Efficient storage of genetic variant data.

1.1 An example

```
##fileformat=VCFv4.2
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS     ID        REF    ALT     QUAL FILTER INFO                              FORMAT      NA00001        NA00002        NA00003
20     14370   rs6054257 G      A       29   PASS   NS=3;DP=14;AF=0.5;DB;H2           GT:GQ:DP:HQ 0|0:48:1:51,51 1|0:48:8:51,51 1/1:43:5:.,
20     17330   .         T      A       3    q10    NS=3;DP=11;AF=0.017               GT:GQ:DP:HQ 0|0:49:3:58,50 0|1:3:5:65,3   0/0:41:3
20     1110696 rs6040355 A      G,T     67   PASS   NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ 1|2:21:6:23,27 2|1:2:0:18,2   2/2:35:4
20     1230237 .         T      .       47   PASS   NS=3;DP=13;AA=T                   GT:GQ:DP:HQ 0|0:54:7:56,60 0|0:48:4:51,51 0/0:61:2
20     1234567 microsat1 GTC    G,GTCT  50   PASS   NS=3;DP=9;AA=G                    GT:GQ:DP    0/1:35:4       0/2:17:2       1/1:40:3
```

GT indicates genotype of individual: 0 means reference allele, 1 means first alternative allele, 2 means second alternative allele, and so on.

Project #5. Efficient storage of genetic variant data.

In this project, you will implement database to store genetic variant data efficiently. After variant calling, genetic variants are often stored in the Variant Call Format (VCF). The current VCF version is 4.2, and you can find its specification here (https://samtools.github.io/hts-specs/VCFv4.2.pdf). In this VCF format, information about genetic variants such as genotypes, genotype quality, and depth is stored as a text file while the text VCF file is often gzipped. VCFTools (https://vcftools.github.io/index.html) is software that researchers often use to work with VCF files. For this project, let's assume we are interested in storing only genotypes (GT filed) of all individuals, ignoring other fields such as GQ and DP. Can you come up with efficient data storage format for this? After you come up with the format, you need to implement conversion from VCF file to your format. In addition, you will need to implement following operations that can be applied to the new format.

Project #5. Efficient storage of genetic variant data.

1. Extract/remove a set of individuals (given as input) from the new format (--keep and --remove options in VCFTools).
2. Extract/remove a set of genetic variants (given as input) from the new format (--snps and --exclude options in VCFTools).
3. Calculate MAF for all genetic variants (--freq option in VCFTools).
4. Find genetic variants that are singletons (only one individual has a genetic variant) and individuals who have those singletons (--singletons option in VCFTools).

Note that you cannot use PLINK binary format for your new storage format because PLINK only allows bi-allelic variants; each genetic variant can have only 2 different alleles (e.g. A and C for SNP 1, C and T for SNP 2, but no A, C, and T for SNP 3). Your storage format would need to allow an arbitrary number of alleles for genetic variants as specified in the VCF specification.

Project #5. Efficient storage of genetic variant data.

To test your program, use the VCF files from 1000 genomes (https://www.internationalgenome.org/data/ Use phase 3 VCF). Compare the performance (the runtime and memory usage) of your program to the performance of VCFTools (using gzipped VCF as input for VCFTools) for the set of 4 operations specified above. You may want to run the same operation multiple times and get average runtime and memory usage. Also, measure how long it takes to convert from the VCF format to your new format.

Project #5. Efficient storage of genetic variant data.

(Graduate students): Please choose one of the following ideas.

Idea 1. In addition to GT, add GQ (genotype quality) and DP (depth) information to your new data format. One operation you need to support is to set genotype to missing (./.) if GQ or DP value is less than a certain threshold (given as input). This is the same as --minGQ and –minDP option in VCFTools. Also, implement --depth, --site-depth, and --site-mean-depth options in VCFTools.

Idea 2. Implement options to convert your new data format to PLINK binary and VCF formats. For PLINK binary format, you will need to remove genetic variants that have more than 2 alleles.