EDA of 1000 Genomes Project: Report on Data and Results

The 1000 Genomes Data comes from the study 1000 Genomes Project that studied the genome variation among globally diverse people and produced an immense catalog of genetic variation among humans. The 1000 Genomes Project provides a thorough description of human genetic variation by whole-genome sequencing to a diverse set of people. This study would not have been possible though without the findings from the Human Genome Project, which set the stage for whole genome sequencing for geneticists by creating the reference genome. The HGP determined humans have around 23,000 genes, where the majority (about half) of the genome is "junk DNA" that do not code for any gene, and the nucleotide sequences between two individuals are nearly 99% identical. However, it is these rare single nucleotide polymorphisms (SNPs) that are responsible for causing diseases and mutations in humans. For the 1000 Genomes Project, Illumina and ABI SOLid DNA sequencing technology is used to generate hundreds of millions to billions of small sequence reads at one time. The 454 method was also used for sequencing DNA fragments equivalent up to a billion bases in a single day. Furthermore, there were technologies used for data collection that consisted of whole- genome sequencing (mean depth = 7.4x), deep exome sequencing (mean depth = 65.7x) and high density SNP microarray genotyping. These developments in sequencing technology sharply reduced the cost of sequencing. The 1000 Genomes Project was the first study to sequence the genomes of a large number of people and to provide a comprehensive resource on human genetic variation.

The data generation process consisted of three different types of collection: low coverage whole-genome sequencing, which sequences small parts of many different sequences and imputes the rest; deep exome sequencing which identifies protein- encoding DNA and sequences that section among many different individuals to understand how that section varies; and density microarray genotyping where fluorescent markers bind to the most common DNA sequence for protein-encoding genes and the markers used to see whether someone has the common sequence or a mutated sequence. This data collected represents a diverse population, with human genomes sequenced from USA, South America, Africa, Europe, East Asian, and Southern Asia. This diversity was crucial for the Project to discover certain significant findings, such as: individuals from African ancestry population harbor the greatest number of variant sites and individuals from recently admixed populations show great variability in the number of variants. This was important since past genetic studies had found a majority of genes to be linked to a rare disease that was prevalent with European ancestry as opposed to African ancestry, but this was mostly due to a bias of current genetic studies and not a bias in demography. The diverse population helped to discover that although most common variants are shared across the world, rarer variants are restricted to closely related populations (86% of variants restricted to a single continental group). However, this is not to say that the population used in this project could not be *more* diverse. For examples, those from populations in Australia, Russia, and the Middle East are not represented in this study, so they would not benefit from any conclusions drawn from later studies that utilize 1000 Genomes Project data to help classify which SNPs contribute to certain diseases. These populations could contain rare SNP variations not seen in any other groups, so their genomes would need to be sequenced in order for any conclusions to apply to those humans.

The 1000 Genomes Data originates from The International Genome Sample Resource's Data portal to access their available data on the 1000 Genomes Project. This data can be downloaded from the FTP site hosted at the EBI, whose server requires a password for registered users or connect as "Guest" to download the FTP data files. However, in my case I am able to ssh onto a DSMLP server provided by data science faculty at UCSD and launch a Docker container to access similar file types and data found through this FTP. The different file types encountered through the 1000 Genomes data processing are .fasta, .fastq, .sam, .bam, and .vcf. Fastq is a file format that contains nucleotide sequence reads with corresponding quality scores, while .fasta file is the reference genome used to map .fastq files. Using software known as "BWA", we can take a .fastq and map it to a .fasta file to produce a .sam file, which is used to represent genome sequencer reads that have been aligned to genome sequences. Transitioning to "samtools" software, we can take our .sam file and create its binary version .bam, while also sorting, indexing, and marking duplicates within the .bam file. To finally reach .vcf file format (otherwise known as Variant Call Format), we can use "GenomeAnalysisToolKit" software function "HaplotypeCaller" in unison with the original .fasta file and the new .bam file to output a .vcf which will contain gene sequence variations. Within the 1000 Genomes data, there exists a .vcf for each chromosome 1-22 that stores position of variation, list of alternative alleles, and the reference base, along with other information.

For the exploratory data analysis replication of filtering, recoding, and merging various chromosome number vcfs, the softwares "plink2" and "bcftools" were useful. To filter and recode the .vcf, plink2 offers various flags that were utilized such as: make-bed, snps-only, maf, geno, mind, and recode. Make-bed creates the files .bed, .bim, and .fam which are important bfiles needed as input for later-down-the-pipeline PCA calculation and visualization. Snps-only excludes variants with one or more multi-character allele codes. Maf filters out variants with a minor allele frequency below a given threshold, and the chosen value of .3 will do well in handling major outliers in the sample while still including SNPs that vary a fair amount across people. Geno filters variants with missing call rates that exceed the given threshold, and the chosen value of .3 effectively gets rid of SNPs with a bunch of missingness while still including enough SNPs with information across the vast majority of samples. Mind does the same as geno, except for individuals above the threshold, and the chosen value of .3 makes sure not to get rid of too many samples unless they have a plethora of missingness, leaving enough people to still run a PCA. Recode creates a text fileset .ped, .map, and specified input "vcf" will also make the plink2 command output a new vcf. These filterings and recoding must be done for each chromosome individually before merging of vcfs, or else it would take a very long time to run the program. The program "bgzip" with flag "-c" takes in a vcf and quickly outputs a gzipped vcf. Next, "bcftools" is used to first create a TBI-formatted index of each gzipped vcf and then concat each vcf to output a merged chromosome vcf. Finally for the last step before PCA can be run on these merged chromosomes, a plink2 command with flag make-bed must be run in order to collect the related bfiles necessary to run the merged chromosomes through PCA.

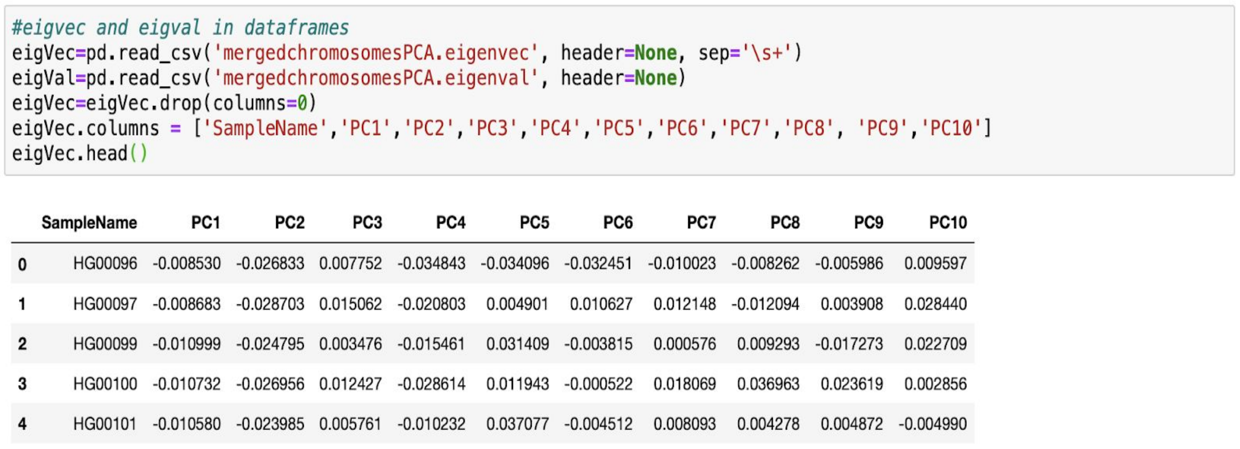
In order to run a successful PCA and visualize the results, plink2 and imported packages such as pandas, matplotlib.pyplot, and seaborn in python are useful in carrying out the rest of the EDA. Taking the merged chromosomes' bfiles, plink2 flag "pca" with specification of number of components outputs two files - one that contains the eigenvalues and another that contains the eigenvectors from the PCA. This data can be read easily into a csv through pandas in order to view what the tables look like. Recoding of the eigenvector's column headers and dropping the duplicate sample names column makes the eigenvector's dataframe more presentable and uncomplicated for later-down-the-pipeline use to illustrate the PCA. In addition, the International Genome Sample Resource provides a tsv that contains information on each sample name's superpopulation code that can be read into a csv using pandas. This table's sample name column can be merged with the eigVec table's sample name column and reindexed to only include all columns from eigVec with just the "SuperpopulationCode" column from igsr. The only population codes given through IGSR are "['EUR', 'EAS', 'AMR', 'SAS', 'AFR']" which stand for European, East Asian, American, South Asian, and African. The reason a PCA is run on these merged chromosomes is simple - this is a method that helps summarize data by taking its characteristics (different variables of data) and finds the best possible characteristics that summarize the data as well as possible. It does this by searching for original variables that show as much variation as possible between samples and grouping redundant properties that are related to produce newly scaled features, otherwise known as principal components, that it believes are the best predictors of the data. Since they are output in order of importance through the plink2 command, principal components 1 and 2 can be plotted (x,y) on a scatter-plot to visualize the results of the best components. Seaborn's scatterplot function where x = PC1, y = PC2, and points colored by superpopulation code, along with matplotlib's plt.legend, results in the visualization shown below.

Based on the plot displayed, we can see that PCA clustered in groups similar to the population code splits. Specifically, those of East Asian and African ancestry are almost grouped entirely into their own clusters based on the PCA plot of component 1 and 2. This means each population forms its own unique cluster, illustrating an underlying connection between members of a specific group and the SNPs found within their genome. In the context of this analysis, it shows how ancestry is linked to one's genetic code and therefore can be predicted through evaluation of which specific SNPs are found. For example, the plot shows those with African ancestry mostly all clustered to the right of the graph, which can be correlated to research on how individuals from recently admixed populations show great variability in the number of variants, roughly proportional to the degree of recent African ancestry in their genomes. Therefore, their component values for PC1, positive values, set them apart from the rest of the samples' component values, mostly all negative values. In addition, it has been shown that those from European, Asian and American populations undergo bottlenecks, which is a massive reduction of population size and subsequently variation resulting in a small population passing on a smaller genetic diversity to the next generation. This could explain why those with superpopulation codes EUR, AMR, and SAS, while in their own clusters, are all clustered near one another or on top of each other.

As with any research study analyzing data, there exist limitations within my approach that could lead to biases in the results. The aforementioned exclusion of populations like Russia, the Middle East, and Australia results in a major bias for future studies that hope to use the 1000 Genomes data to form conclusions. For instance, if geneticists wanted to perform a GWAS (Genome-Wide Association Study) to associate specific genetic variations to particular diseases, their results can only apply to the specific population used in the data analyzation process. The data gathered by 1000 Genomes would not be applicable to people from Russia or Australia due to their lack of representation and differences between populations of which SNPs correlate to a trait. Another bias occurs when I filter the SNPs based on boundaries set to dispose of outliers in the data. While this helps to rid of false positives within the SNPs, it can also cause some important SNPs to be missed and limits the power of association, or strength or relationships between variables later down the analysis. These biases can be addressed better in the future by including a wider diversity of population samples and lessening the thresholds to filter the SNPs.

In conclusion, an exploratory data analysis was run on data gathered from the 1000 Genomes Project to produce a scatterplot visual of principal components calculated through PCA. The softwares bwa, samtools, and gatk were useful in converting a .fastq file to a .bam file and to a .vcf file. The softwares plink2, bcftools, and gatk were helpful in filtering, merging, and running a PCA on .vcf chromosomes 20-22. To visualize the output from the PCA, I used the programming language Python with the packages pandas, seasborn, and matplotlib.pyplot to create a scatterplot graph of PCA components 1 and 2, with points color-grouped by their superpopulation code. The graph illustrates how the PCA clusterings correspond to the clusterings created by the superpopulation codes, which exhibits the underlying connection between the SNPs found on a person's genome and their ancestry. The 1000 Genomes Project paved the way for geneticists to conduct GWASs and locate SNPs that are highly correlated to a trait or illness, yet there is still much more to be found within the realm of genetics which I will continue to explore further next quarter!

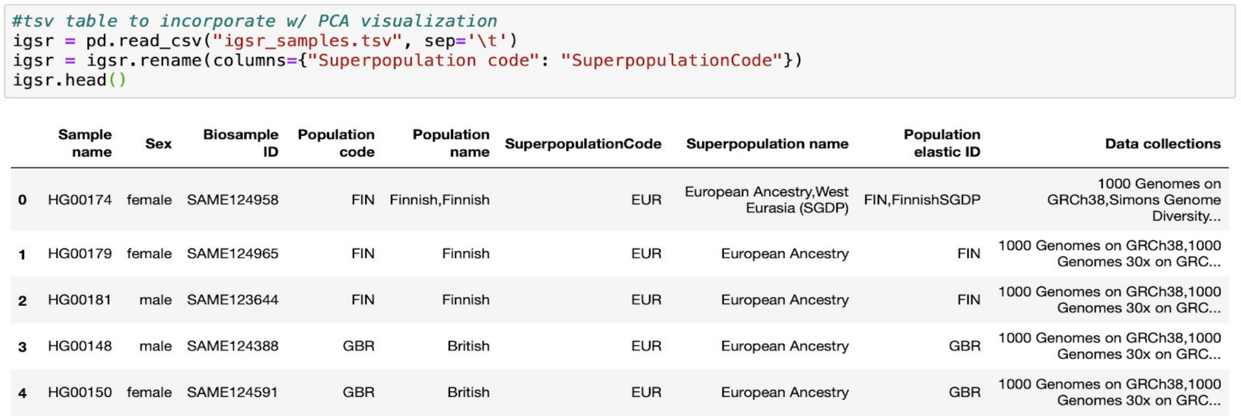
eigVec Table:



```
#eigvec and eigval in dataframes
eigVec=pd.read_csv('mergedchromosomesPCA.eigenvec', header=None, sep='\s+')
eigVal=pd.read_csv('mergedchromosomesPCA.eigenval', header=None)
eigVec=eigVec.drop(columns=0)
eigVec.columns = ['SampleName','PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8', 'PC9','PC10']
eigVec.head()
```

	SampleName	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
0	HG00096	-0.008530	-0.026833	0.007752	-0.034843	-0.034096	-0.032451	-0.010023	-0.008262	-0.005986	0.009597
1	HG00097	-0.008683	-0.028703	0.015062	-0.020803	0.004901	0.010627	0.012148	-0.012094	0.003908	0.028440
2	HG00099	-0.010999	-0.024795	0.003476	-0.015461	0.031409	-0.003815	0.000576	0.009293	-0.017273	0.022709
3	HG00100	-0.010732	-0.026956	0.012427	-0.028614	0.011943	-0.000522	0.018069	0.036963	0.023619	0.002856
4	HG00101	-0.010580	-0.023985	0.005761	-0.010232	0.037077	-0.004512	0.008093	0.004278	0.004872	-0.004990

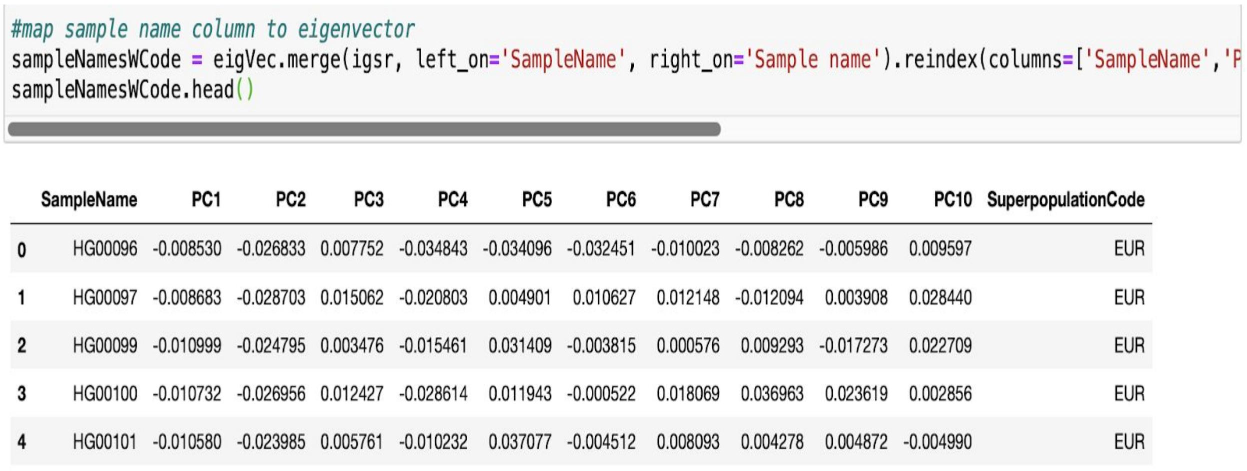
igsr Table:



```
#tsv table to incorporate w/ PCA visualization
igsr = pd.read_csv("igsr_samples.tsv", sep='\t')
igsr = igsr.rename(columns={"Superpopulation code": "SuperpopulationCode"})
igsr.head()
```

	Sample name	Sex	Biosample ID	Population code	Population name	SuperpopulationCode	Superpopulation name	Population elastic ID	Data collections
0	HG00174	female	SAME124958	FIN	Finnish,Finnish	EUR	European Ancestry,West Eurasia (SGDP)	FIN,FinnishSGDP	1000 Genomes on GRCh38,Simons Genome Diversity...
1	HG00179	female	SAME124965	FIN	Finnish	EUR	European Ancestry	FIN	1000 Genomes on GRCh38,1000 Genomes 30x on GRC...
2	HG00181	male	SAME123644	FIN	Finnish	EUR	European Ancestry	FIN	1000 Genomes on GRCh38,1000 Genomes 30x on GRC...
3	HG00148	male	SAME124388	GBR	British	EUR	European Ancestry	GBR	1000 Genomes on GRCh38,1000 Genomes 30x on GRC...
4	HG00150	female	SAME124591	GBR	British	EUR	European Ancestry	GBR	1000 Genomes on GRCh38,1000 Genomes 30x on GRC...

sampleNameWCode Table:



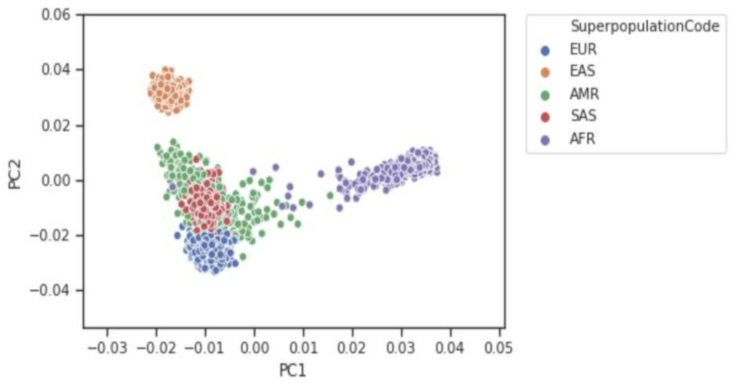
```
#map sample name column to eigenvector
sampleNamesWCode = eigVec.merge(igsr, left_on='SampleName', right_on='Sample name').reindex(columns=['SampleName','P
sampleNamesWCode.head()
```

	SampleName	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	SuperpopulationCode
0	HG00096	-0.008530	-0.026833	0.007752	-0.034843	-0.034096	-0.032451	-0.010023	-0.008262	-0.005986	0.009597	EUR
1	HG00097	-0.008683	-0.028703	0.015062	-0.020803	0.004901	0.010627	0.012148	-0.012094	0.003908	0.028440	EUR
2	HG00099	-0.010999	-0.024795	0.003476	-0.015461	0.031409	-0.003815	0.000576	0.009293	-0.017273	0.022709	EUR
3	HG00100	-0.010732	-0.026956	0.012427	-0.028614	0.011943	-0.000522	0.018069	0.036963	0.023619	0.002856	EUR
4	HG00101	-0.010580	-0.023985	0.005761	-0.010232	0.037077	-0.004512	0.008093	0.004278	0.004872	-0.004990	EUR

PCA Plot:

```
#plot scatterplot of PC1 and PC2 grouped by SuperpopulationCode
seaborn.set(style='ticks')
fg = seaborn.scatterplot(data=sampleNamesWCode, x='PC1', y='PC2', hue='SuperpopulationCode', hue_order=list(populati
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

```
<matplotlib.legend.Legend at 0x7facee2329e8>
```


SuperpopulationCode
EUR
EAS
AMR
SAS
AFR
PC2
PC1