Intro to popgen

Introduction to Population Genetics, Genetic Diversity, and SNPs

Population genetics is the study of genetic differences within and between populations. It focuses on understanding how these differences arise, how they are distributed, and their role in evolutionary processes. Key areas of research include **adaptation**, **speciation**, and **population structure**.

Genetic Diversity: The Foundation of Evolution

Genetic diversity refers to the variation in DNA sequences among individuals in a population. It plays a critical role in:

1. **Adaptation**: High genetic diversity increases the likelihood that some individuals in a population can survive and reproduce under changing environmental conditions.
2. **Evolutionary potential**: Diversity provides the raw material for natural selection, enabling populations to evolve over time.

The DNA of any two humans is 99.6% identical, while the remaining 0.4%—approximately **12 million base pairs**—accounts for individual differences. These variations can be neutral, beneficial, or detrimental, depending on their context in the genome.

Single Nucleotide Polymorphisms (SNPs)

SNPs are the most common type of genetic variation and represent a change in a single nucleotide (A, C, T, or G).

- **SNP**: Found in >1% of the population.
- **Mutation**: Found in <1% of the population.

Key Characteristics of SNPs:

1. **Frequency**: SNPs account for **90% of human genetic variation**.
2. **Distribution**: They occur approximately every **300 to 1000 base pairs** in the genome.
3. **Binarity**: SNPs are typically binary, meaning there are two possible nucleotides at a given position.

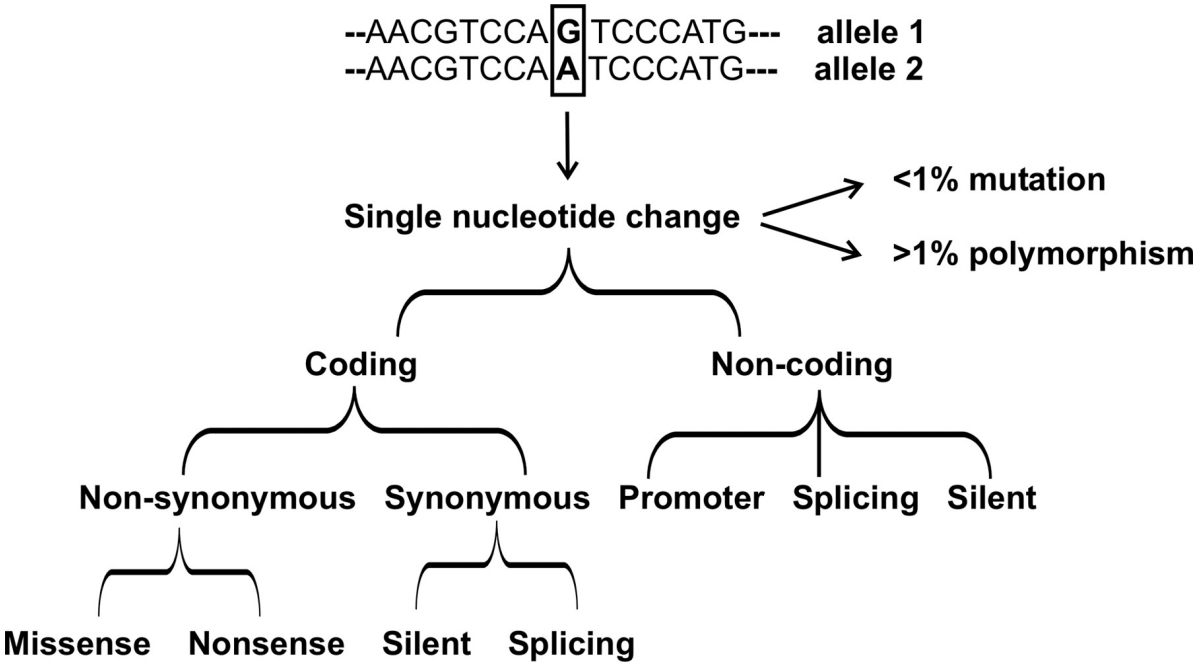
Allele Types at SNP Loci:

- **Major allele**: The nucleotide found in **more than 50%** of the population.
- **Minor allele**: The nucleotide found in **less than 50%** of the population.

SNPs in Coding and Non-Coding Regions

SNPs can occur in both **non-coding** and **coding regions**:

1. **Non-coding regions**: SNPs in these regions can affect gene regulation, splicing, or the stability of RNA.
2. **Coding regions**: SNPs in these regions may be:
    1. **Synonymous**: No change in the amino acid sequence due to redundancy in the genetic code.
    2. **Nonsynonymous**: A change in the amino acid sequence, potentially affecting the structure and function of the protein.



SNPs as Tools in Research

Due to their abundance and ease of genotyping, SNPs have become the primary markers in **association studies**, which examine how genetic variations are linked to traits such as:

- **Diseases**: Identifying genetic risk factors for conditions like diabetes or cancer.
- **Phenotypes**: Understanding the genetic basis of traits like height, skin color, or metabolic differences.

The Importance of Studying SNPs

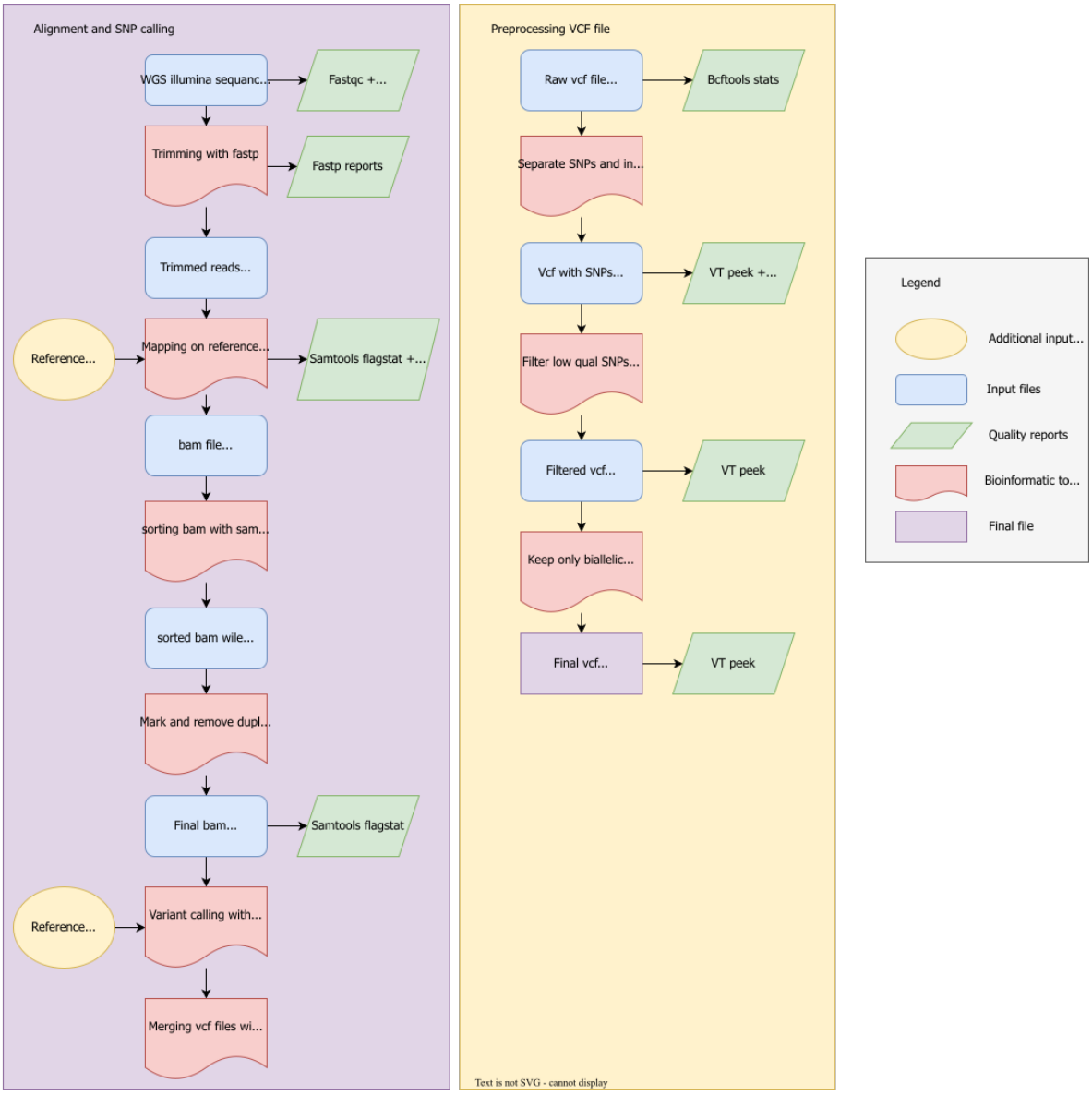
SNPs are invaluable for population genetics and genomics because they:

- **Trace ancestry and population history**: SNP patterns reveal migration, admixture, and evolutionary history.
- **Aid conservation biology**: Assess genetic diversity in endangered species to guide conservation efforts.
- **Facilitate personalized medicine**: Help identify genetic variants associated with drug responses and diseases.

By exploring genetic diversity and SNPs, population genetics provides a powerful framework for understanding evolution, human health, and biodiversity.

Bioinformatic pipeline to get SNPs

This is schematic pipeline to get final merged and filtered VCF file.



From now on we will work in Plink to do some analysis and prepeare data for R.

We will use Plink in Linux terminal. If you don't know how to use Linux please take a look into basic Linux commands:

Basic Linux commands

Biallelic Variants in VCF Files for PLINK

When preparing VCF (Variant Call Format) files for PLINK analysis, it's crucial to understand and focus on biallelic variants. Here's what you need to know:

What are Biallelic Variants?

Biallelic variants are genetic variations where only two alleles (versions of a gene) are observed at a specific locus (position) in a population. In the context of SNPs (Single Nucleotide Polymorphisms), this means there are only two possible nucleotides at a given position.

Why Biallelic Variants for PLINK?

- **Simplicity:** PLINK is designed to work primarily with biallelic variants, as they are easier to analyze statistically.
- **Common in SNPs:** Most SNPs are naturally biallelic, making them ideal for population genetics studies.
- **Computational Efficiency:** Biallelic variants require less computational resources to process and analyze.

Preparing VCF Files for PLINK

When preparing your VCF file for PLINK analysis, ensure that:

1. **Filter for Biallelic Sites:** Remove any multiallelic variants (sites with more than two alleles) from your VCF file.
2. **Check the REF and ALT Columns:** In your VCF file, the REF (reference) column should contain one allele, and the ALT (alternate) column should contain only one alternate allele.
3. **Example:** A valid biallelic entry might look like this in your VCF: `chr1 100 . A G . PASS .` Here, 'A' is the reference allele, and 'G' is the single alternate allele.

Important Considerations

- **Quality Control:** Ensure your variants pass appropriate quality filters before including them in your PLINK analysis.
- **Indels:** While PLINK can handle biallelic insertions and deletions (indels), some analyses might be more straightforward with SNPs only.
- **Documentation:** Always document any filtering steps you perform on your VCF file before PLINK analysis.

By focusing on biallelic variants in your VCF file, you'll ensure compatibility with PLINK and set a solid foundation for your population genetics analyses.

Plink

We know two versions of Plink that are commonly used:

- Plink 1.9 - Plink 19 manual
- Plink 2.0 - Plink 2 manual

They are quite different specially with some functions that calculate matrices. We will use Plink 2, because some calculations that are available just in Plink 1 directly in R in our next sessions.

You need to download `plink2.exe` file from official website. If you are using Linux download `plink2` file. Place this file in working folder when you already have map and ped file.

Input files for Plink

`.ped` file:

File (pedigree) that contains data about SNPs for all individuals together with some metadata about individuals.

Column	1	2	3	4	5	6	7->
Data	Family ID	Individual ID	Paternal ID	Maternal ID	Sex	Phenotype	Genotypes
Description	Group of individual	ID for sample or individual	ID from father	ID from mother	Sex	Phenotype of individual	SNPs data (2 for each marker)
Values	custom	custom	0=no data	0=no data	1=M 2=F 0=unknown	0=unknown 1=unafected 2=affected	A C G T / 1 2 3 4 / 1 2 = alel 0=missing

Example of a PED file of the standard PLINK format:

```
FAM1   NA06985  0 0 1 1 A T T T G G C C A T T T G G C C
FAM1   NA06991  0 0 1 1 C T T T G G C C C T T T G G C C
  0    NA06993  0 0 1 1 C T T T G G C T C T T T G G C T
  0    NA06994  0 0 1 1 C T T T G G C C C T T T G G C C
  0    NA07000  0 0 2 1 C T T T G G C T C T T T G G C T
  0    NA07019  0 0 1 1 C T T T G G C C C T T T G G C C
  0    NA07022  0 0 2 1 C T T T G G 0 0 C T T T G G 0 0
  0    NA07029  0 0 1 1 C T T T G G C C C T T T G G C C
FAM2   NA07056  0 0 0 2 C T T T A G C T C T T T A G C T
FAM2   NA07345  0 0 1 1 C T T T G G C C C T T T G G C C
```

`.map` file:

File that contains data about SNPs and their positions on chromosomes.

Column	1	2	3	4
Data	Chr	SNP ID	cM	Base pair position
Description	Chromosome id where the SNP is	ID of SNP	Distance in centimorgan (genetic distance)	SNP position

Example of a MAP file of the standard PLINK format:

```
21    rs11511647    0    26765
 X     rs3883674    0    32380
 X    rs12218882    0    48172
 9    rs10904045    0    48426
 9    rs10751931    0    49949
 8    rs11252127    0    52087
10    rs12775203    0    52277
 8    rs12255619    0    52481
```

If we already have `.map` and `.ped` file:

Because we usually work with big files that contains SNPs across whole genome it is suggest to convert files to binary files:

```
./plink2.exe --ped hapmap1.ped --map hapmap1.map --make-pgen --out hapmap_bin
```

This command will create three binary files:

- pgen
- psam
- pvar

Filtering

- Filter missing Genotypes: Removes variants and samples with >5% missing data.

```
./plink2.exe --pfile hapmap_bin --geno 0.05 --mind 0.05 --make-pgen --out hapmap1_filtered
```

Removes variants and samples with >5% missing data.

- Filter by MAF

```
./plink2.exe --pfile hapmap1_filtered_geno_mind --maf 0.05 --make-pgen --out hapmap1_filte
```

Retains variants with a Minor Allele Frequency (MAF) > 5%.

- Filter by HWE

```
./plink2.exe --pfile hapmap1_filtered_maf --hwe 1e-6 --make-pgen --out hapmap1_filtered_hw
```

- Filter by chromosome

```
./plink2.exe --pfile hapmap1_filtered_hwe --chr 1 --make-pgen --out hapmap1_chr1
```

This keeps just SNPs on chromosome 1. Be careful we will use whole dataset, so fo next filtering we will use output from previous step.

- LD prunning

```
./plink2.exe --pfile hapmap1_filtered_hwe --indep-pairwise 50 5 0.2 --make-pgen --out hapm
```

Prunes SNPs in high linkage disequilibrium (LD) for downstream analyses.

- We can do all filtering in just one step. But keep track what was used for filtering:

```
./plink2.exe --pfile hapmap_bin --geno 0.05 --mind 0.05 --maf 0.05 --hwe 1e-6 --indep
```

Statistics

We can calculate many statistics also with Plink, but visualizations need to be done in R. So we will do all of them also in R with some specific R packages. This features are important specially for large dataset, because we can run Plink in Linux servers and then outputs are text files which we can just import into R.

- Missingness

```
./plink2.exe --pfile hapmap1_filtered_hwe --missing --out missingness
```

- Allele frequency

```
./plink2.exe --pfile hapmap1_filtered_hwe --freq --out allele_freq
```

- Hardy Weinberg

```
./plink2.exe --pfile hapmap1_filtered_hwe --hardy --out hardy
```

- PCA (principal component analysis)

```
./plink2.exe --pfile hapmap1_filtered_LD --pca 20 --make-pgen --out pca
```

Computes the first 20 principal components to investigate population structure.

- MDS (multidimensional scaling) → just on plink 1!!!

```
./plink2.exe --pfile hapmap1_filtered_LD --cluster --mds-plot 4 --make-pgen --out mds

./plink2.exe --pfile hapmap1_filtered_LD --cluster --matrix --make-pgen --out mds
```

Performs multidimensional scaling to identify population substructure.

- Pairwise relationship matrix

```
./plink2.exe --pfile hapmap1_filtered_LD --make-rel square --out rel_matrix
```

```
./plink2.exe --pfile hapmap1_filtered_LD --make-king-table --out king_matrix
```

- Heterozygosity

```
./plink2.exe --pfile hapmap1_filtered_hwe --het --out het_results
```

Plink in R

Here I will show usage of Plink 1.9, so that you will have some comparison.

In the first part of the exercises, we introduced PLINK and the command line, which allows us to execute commands. In the previous exercises, we learned about R, RStudio, and RMarkdown. R allows us to run commands from the command line using the `system()` function.

```
{r}
system("plink/plink --file plink/hapmap1 --make-bed --out plink/hapmap1_bin")
```

R allows us to program a function where we can specify arguments without typing `plink` beforehand. This is useful if we have the `plink.exe` file stored elsewhere, not in the same folder as the data files. In such cases, the command would look like this:

```
{r}
system("C:/Users/Lenovo/Documents/R/EPG/EPG_quarto_book/plink/plink --file plink/hapmap1 -
```

This approach is not as clear. Instead, we'll create a function `runPLINK` using the `function()` command that takes the arguments and expression. Our argument will be `PLINKoptions`, and the expression will be `paste("plink", PLINKoptions)`.

```
{r}
runPLINK <- function(PLINKoptions = "") system(paste("plink/plink", PLINKoptions))
```

The function we created will appear in the **Environment** pane, and it will require the `PLINKoptions`, such as `--file hapmap1...`.

To run it, you simply call:

```
{r}
runPLINK("--file plink/hapmap1 --make-bed --recode --out plink/hapmap3_bin")
```

Visualization of the results in R

Load needed R packages:

```
library(tidyverse)
```

```
Warning: package 'ggplot2' was built under R version 4.3.3
```

```
library(reshape2)
```

Hardy Weinberg

```
#Before import delete # sign from header
hardy_stat <- read.table("plink/hardy.hardy", header = TRUE)
head(hardy_stat)
```

```
  CHROM         ID A1 AX HOM_A1_CT HET_A1_CT TWO_AX_CT O.HET_A1. E.HET_A1.
1     1  rs6681049  2  1        59        22         8  0.247191  0.335816
2     1  rs4074137  2  1        76        12         1  0.134831  0.144931
3     1  rs1891905  2  1        35        36        18  0.404494  0.481757
4     1  rs9729550  2  1        69        17         3  0.191011  0.225035
5     1 rs12044597  2  1        18        54        17  0.606742  0.499937
6     1 rs10907185  2  1        45        37         7  0.415730  0.408850
          P
1 0.0218819
2 0.4252430
3 0.1286230
4 0.1508470
5 0.0581590
6 1.0000000
```

Column	Description
CHROM	Chromosome number where the SNP is located.
ID	Variant identifier (SNP ID or combination of chromosome:position:REF).
A1	Allele 1, which is the reference allele for the test.
AX	Total number of unique alleles (usually 2 for bi-allelic SNPs).
HOM_A1_CT	Count of individuals homozygous for Allele 1.
HET_A1_CT	Count of heterozygous individuals (carrying both Allele 1 and the alternate allele).
TWO_AX_CT	Count of individuals homozygous for the alternate allele (Allele 2).
O(HET_A1)	Observed heterozygosity, calculated as: HET_A1_CT / Total_Genotypes
E(HET_A1)	Expected heterozygosity under Hardy-Weinberg Equilibrium, based on allele frequencies.
P	p-value from the Hardy-Weinberg Equilibrium test (indicates how well the SNP fits HWE expectations).

Calculate mean value of expected and observed heterozygosity per each chromosome:

```
mean_hardy <- group_by(hardy_stat, CHROM) %>%
  summarise(
    mean_O_HET = mean(O.HET_A1., na.rm = TRUE),
    mean_E_HET = mean(E.HET_A1., na.rm = TRUE),
    mean_P = mean(P, na.rm = TRUE)
  )

mean_hardy
```

```
# A tibble: 22 x 4
   CHROM mean_O_HET mean_E_HET mean_P
   <int>      <dbl>      <dbl>  <dbl>
 1     1      0.357      0.356  0.600
 2     2      0.356      0.355  0.610
 3     3      0.356      0.357  0.601
 4     4      0.358      0.358  0.596
 5     5      0.358      0.358  0.617
 6     6      0.359      0.358  0.601
 7     7      0.355      0.355  0.604
 8     8      0.374      0.373  0.612
 9     9      0.360      0.359  0.616
10    10      0.359      0.359  0.602
# i 12 more rows
```

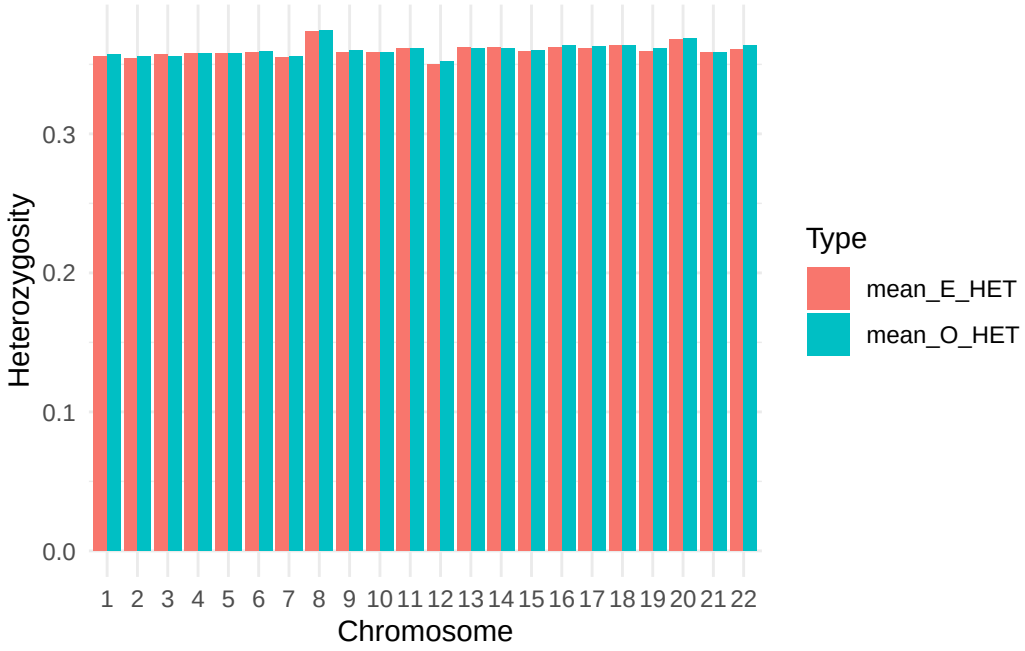
Plotting the expected and observed heterozygosity side by side with barplot:

```
# Reshaping data into long format
mean_hardy_long <- mean_hardy %>%
  select(CHROM, mean_E_HET, mean_O_HET) %>%
  pivot_longer(cols = starts_with("mean"),
               names_to = "type",
               values_to = "value")

head(mean_hardy_long)
```

```
# A tibble: 6 x 3
  CHROM type       value
  <int> <chr>      <dbl>
1     1 mean_E_HET 0.356
2     1 mean_O_HET 0.357
3     2 mean_E_HET 0.355
4     2 mean_O_HET 0.356
5     3 mean_E_HET 0.357
6     3 mean_O_HET 0.356
```

```
# Plotting using ggplot
ggplot(mean_hardy_long, aes(x = factor(CHROM), y = value, fill = type)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Chromosome", y = "Heterozygosity", fill = "Type") +
  theme_minimal()
```



Here we can see, that the difference between both values is really small. Usually We want observed heterozygosity to be higher, beacuse this means that we have more heterozygotes inour population than expected. If the observed value is smaller than expected this can indicate that we have inbreeding in our population.

PCA

```
#Before import delete # sign from header
pca_df <- read.table("plink/pca.eigenvec", header = TRUE)
head(pca_df)
```

```
     FID IID       PC1         PC2         PC3         PC4          PC5          PC6
1 HCB181   1 0.1085610  0.0260607 -0.0924137 -0.0343485 -0.10610900  0.04450110
2 HCB182   1 0.1182700 -0.0501544 -0.0611786  0.0242902 -0.05180630  0.13499200
3 HCB183   1 0.1070180  0.1063170 -0.0393811 -0.0189775  0.00440272  0.00081924
4 HCB184   1 0.0993545 -0.0857655 -0.0593316  0.0619104  0.19478200 -0.27069500
5 HCB185   1 0.1067410  0.1728340  0.0528158 -0.1640030 -0.14533000 -0.06500940
6 HCB186   1 0.1129780  0.0478710 -0.0470561  0.1210350  0.02505350  0.17764500
          PC7         PC8         PC9        PC10         PC11        PC12
1 -0.13051400  0.2003240 -0.0347337  0.2535660 -0.00248808  0.0356898
```

```
2  0.11245700  0.0965491 -0.0985641  0.1767950  0.03634590 -0.0126869
3 -0.10302200 -0.0326958 -0.1445300 -0.0517589 -0.00199790  0.1079670
4  0.09247320  0.0347238  0.0832056  0.1062530  0.07672770  0.1034830
5 -0.00416153  0.0774073  0.0202584 -0.0930361  0.04735280  0.1386720
6  0.11951600 -0.0127285  0.1408690 -0.0453216 -0.00914330  0.0803249
        PC13         PC14       PC15       PC16       PC17       PC18
1  0.1059130 -0.074566300 -0.1814910 -0.2333100  0.0591321 0.00560641
2  0.0215172 -0.056746400  0.0325433  0.1460500 -0.0975274 0.12491300
3 -0.2096070  0.006228050  0.1822140 -0.0771186 -0.1286890 0.13830200
4  0.0423965  0.141625000  0.0147856 -0.0434972  0.0104450 0.01873720
5  0.0637130 -0.000376912 -0.1259270 -0.0634153  0.1603430 0.01028080
6 -0.0630175  0.076162300  0.0725746  0.1337940  0.1898460 0.02167050
         PC19       PC20
1  0.03499650  0.2239170
2 -0.02035160  0.1029010
3  0.14445300 -0.1143120
4  0.00335798  0.1748390
5  0.25865500 -0.0632465
6 -0.09704560  0.0815970
```

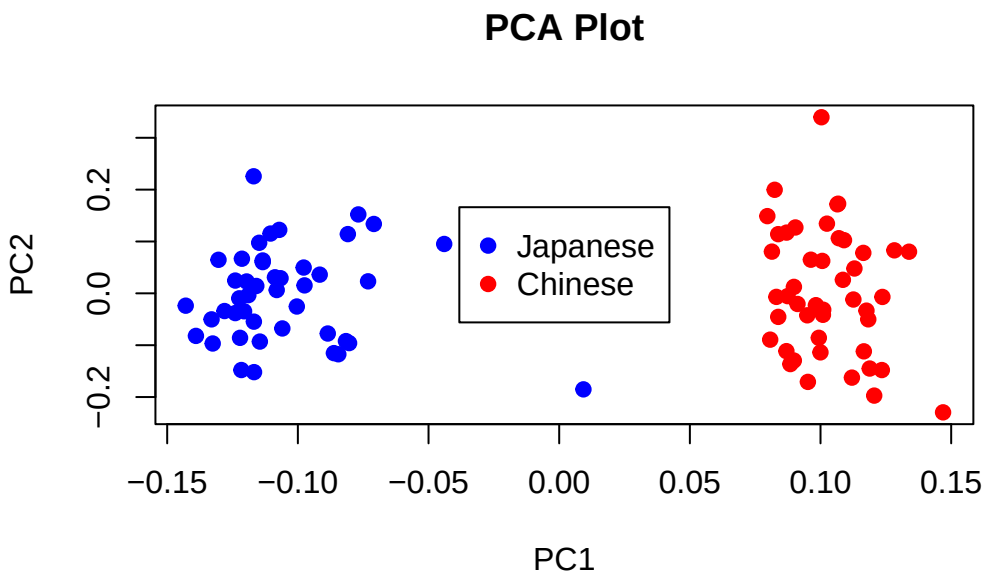
Plot PCA

```r
# Categorize samples based on FID
pca_df$Group <- ifelse(startsWith(pca_df$FID, "HCB"), "Chinese", "Japanese")

# Define colors for groups
colors <- c("Japanese" = "blue", "Chinese" = "red")

# Plot PC1 vs PC2
plot(pca_df$PC1, pca_df$PC2, col=colors[pca_df$Group], pch=20, cex=1.5,
     xlab="PC1", ylab="PC2", main="PCA Plot")

# Add legend
legend("center", legend=names(colors), col=colors, pch=20, pt.cex=1.5)
```



Allele frequency

```
all_freq <- read.table("plink/allele_freq.afreq", header = TRUE)
head(all_freq)
```

```
  CHROM         ID REF ALT PROVISIONAL_REF. ALT_FREQS OBS_CT
1     1  rs6681049   2   1                Y 0.2134830    178
2     1  rs4074137   2   1                Y 0.0786517    178
3     1  rs1891905   2   1                Y 0.4044940    178
4     1  rs9729550   2   1                Y 0.1292130    178
5     1 rs12044597   2   1                Y 0.4943820    178
6     1 rs10907185   2   1                Y 0.2865170    178
```

```
mean_all_freq <- group_by(all_freq, CHROM) %>%
  summarise(mean_all_freq = mean(all_freq$ALT_FREQS, na.rm = TRUE))

mean_all_freq
```

```
# A tibble: 22 x 2
```

```
   CHROM mean_all_freq
   <int>         <dbl>
 1     1         0.270
 2     2         0.270
 3     3         0.270
 4     4         0.270
 5     5         0.270
 6     6         0.270
 7     7         0.270
 8     8         0.270
 9     9         0.270
10    10         0.270
# i 12 more rows
```

IBS matrix

King-algorithm

KING-robust kinship estimator

https://www.cog-genomics.org/plink/2.0/distance

The relationship matrix computed by --make-rel/--make-grm-list/--make-grm-bin can be used to reliably identify close relations within a single population, if your MAFs are decent. However, Manichaikul et al.'s KING-robust estimator can also be mostly trusted on mixed-population datasets (with one uncommon exception noted below), and doesn't require MAFs at all. Therefore, we have added this computation to PLINK 2, and the relationship-based pruner is now based on KING-robust.

The exception is that KING-robust underestimates kinship when the parents are from very different populations. You may want to have some special handling of this case; --pca can help detect it.

Note that KING kinship coefficients are scaled such that duplicate samples have kinship 0.5, not 1. First-degree relations (parent-child, full siblings) correspond to ~0.25, second-degree relations correspond to ~0.125, etc. It is conventional to use a cutoff of ~0.354 (the geometric mean of 0.5 and 0.25) to screen for monozygotic twins and duplicate samples, ~0.177 to add first-degree relations, etc.

Other explanation

The `king` matrix from PLINK is a result of the KING algorithm, which calculates pairwise kinship coefficients between individuals in a dataset. The KING kinship matrix is particularly useful for identifying relatedness and inferring family structures.

Description of the `king` Matrix

1. **Rows and Columns**:
   - Each row and column corresponds to an individual in your dataset.
   - The matrix is symmetric.
2. **Values**:
   - The values represent the kinship coefficient between pairs of individuals.
   - Kinship coefficients can be interpreted as follows:
     - **0.5**: Full siblings or parent-offspring.
     - **0.25**: Half-siblings, avuncular relationships (e.g., uncle-niece), or grandparent-grandchild.
     - **0.125**: First cousins.
     - **0**: Unrelated individuals (in ideal scenarios).
   - Negative values may occur due to population structure or noise, but these are generally treated as zero.

We can use IBS or Kinship values

```
king_mat <- read.table("plink/king_matrix.kin0", header = TRUE)
head(king_mat)
```

```
    FID1 IID1   FID2 IID2  NSNP   HETHET      IBS0      KINSHIP
1 HCB182    1 HCB181    1 56792 0.147961 0.0729152  0.001874760
2 HCB183    1 HCB181    1 57208 0.146745 0.0766501 -0.010755300
3 HCB183    1 HCB182    1 56934 0.146222 0.0718903  0.000525852
4 HCB184    1 HCB181    1 57235 0.144509 0.0732244 -0.004750770
5 HCB184    1 HCB182    1 56964 0.143968 0.0720279 -0.003515610
6 HCB184    1 HCB183    1 57404 0.144084 0.0724166 -0.001372430
```

Column Descriptions

1. **`FID1` and `IID1`**:
   - Family ID and Individual ID of the first individual in the pair.
2. **`FID2` and `IID2`**:
   - Family ID and Individual ID of the second individual in the pair.
3. **`NSNP`**:
   - Number of SNPs used to calculate the relatedness metrics for this pair. Higher values indicate more reliable estimates.
4. **`HETHET`**:
   - Proportion of SNPs where both individuals are heterozygous (used for quality control).
5. **`IBS0`**:
   - Proportion of SNPs where both individuals share zero alleles identical by state (IBS). Higher values often suggest unrelatedness.
6. **`KINSHIP`**:
   - Kinship coefficient, the primary metric indicating the degree of relatedness:
     - $>$ **0.354**: Duplicate or monozygotic twins.
     - **0.177–0.354**: First-degree relatives (parent-offspring, full siblings).
     - **0.0884–0.177**: Second-degree relatives (half-siblings, grandparents, etc.).
     - **0.0442–0.0884**: Third-degree relatives (first cousins).
     - $<$ **0.0442**: Unrelated (or very distantly related).

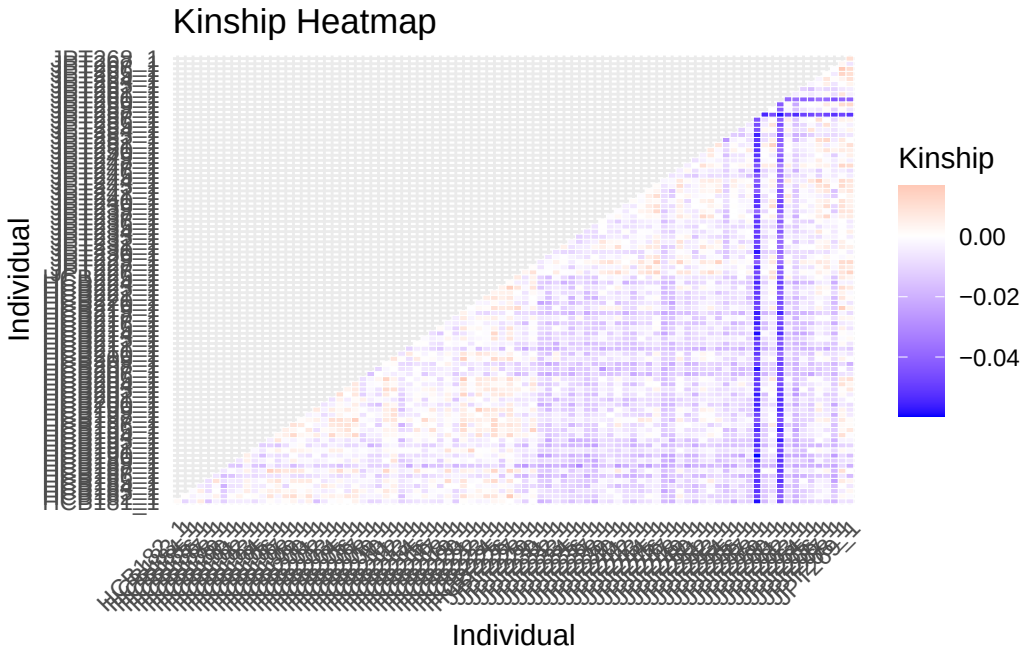
```
# Create a combined ID for both individuals
king_mat$ID1 <- paste(king_mat$FID1, king_mat$IID1, sep = "_")
king_mat$ID2 <- paste(king_mat$FID2, king_mat$IID2, sep = "_")

# Convert to square matrix format
king_matrix <- dcast(king_mat, ID1 ~ ID2, value.var = "KINSHIP")
rownames(king_matrix) <- king_matrix$ID1
king_matrix$ID1 <- NULL
#head(king_matrix)
```

Heatmap

```
# Melt the matrix for ggplot
king_long <- melt(as.matrix(king_matrix), na.rm = TRUE)

# Create the heatmap
ggplot(king_long, aes(Var1, Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", midpoint = 0, name = "Kinship") +
  labs(x = "Individual", y = "Individual", title = "Kinship Heatmap") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



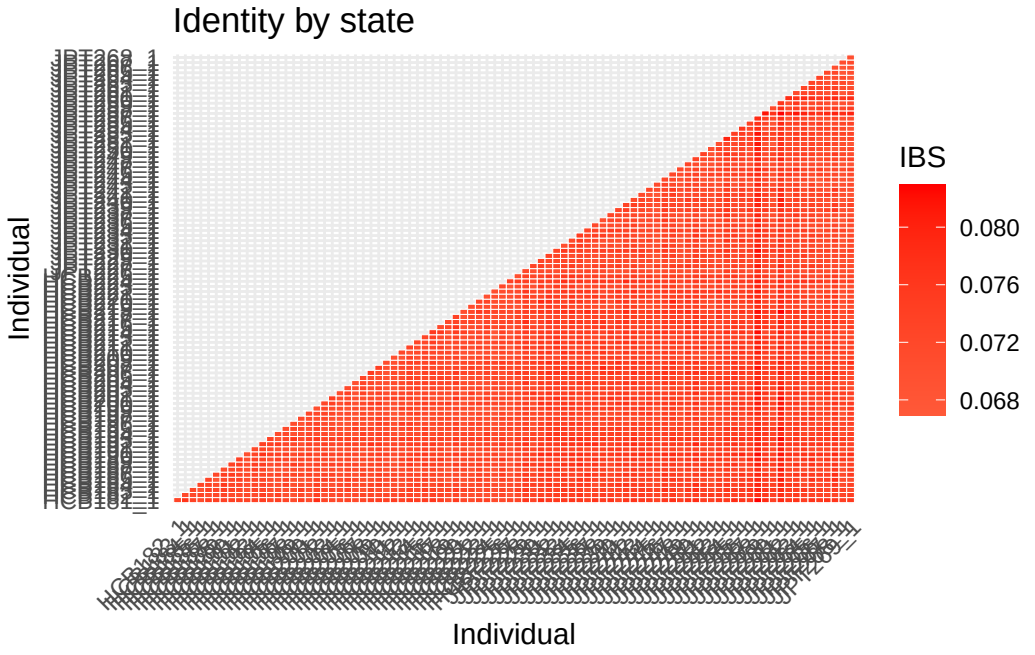
IBS

```
# Convert to square matrix format
king_matrix_ibs <- dcast(king_mat, ID1 ~ ID2, value.var = "IBS0")
rownames(king_matrix_ibs) <- king_matrix_ibs$ID1
king_matrix_ibs$ID1 <- NULL
#head(king_matrix_ibs)
```

Heatmap

```r
# Melt the matrix for ggplot
king_long_IBS <- melt(as.matrix(king_matrix_ibs), na.rm = TRUE)

# Create the heatmap
ggplot(king_long_IBS, aes(Var1, Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", midpoint = 0, name = "IBS") +
  labs(x = "Individual", y = "Individual", title = "Identity by state") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Using `KINSHIP` for the Heatmap

- **What it Represents**:
  - The `KINSHIP` coefficient quantifies relatedness between pairs of individuals.
  - Higher values indicate closer biological relationships:
    - $>$ **0.354**: Monozygotic twins or duplicate samples.
    - **0.177–0.354**: First-degree relatives (parent-offspring, full siblings).

    - **0.0884–0.177**: Second-degree relatives (half-siblings, grandparents, etc.).
    - **0.0442–0.0884**: Third-degree relatives (first cousins).
    - $<$ **0.0442**: Unrelated or very distantly related individuals.

- **Heatmap Insight**:
  - A heatmap of `KINSHIP` will visually group individuals with close familial ties.
  - Clusters of high values can indicate families or genetic subpopulations.

- **Example Interpretation**:
  - Dark red (high `KINSHIP`) between two individuals suggests they are closely related, like siblings or parent-offspring.
  - Light colors or near-zero values indicate unrelated individuals.

Using `IBS0` for the Heatmap

- **What it Represents**:
  - The `IBS0` (Identity by State 0) value measures the proportion of SNPs where two individuals share **zero alleles** identical by state.
  - Higher `IBS0` values indicate **less relatedness**, while lower values suggest closer relationships.

- **Heatmap Insight**:
  - A heatmap of `IBS0` will highlight unrelated pairs with high values.
  - Clusters of low `IBS0` values represent related individuals.

- **Example Interpretation**:
  - Dark red (low `IBS0`) indicates that two individuals are more likely related.
  - Light colors (high `IBS0`) suggest no shared ancestry or distant relationship.

Comparison of Results

Metric	High Value Interpretation	Low Value Interpretation	Best Use Case
`KINSHIP`	Very closely related individuals (e.g., twins, siblings).	Unrelated individuals.	Visualizing clusters of related individuals (families, subpopulations).
`IBS0`	Distantly related or unrelated individuals.	Closely related individuals (fewer SNPs where both share 0 alleles).	Checking for distant versus close relationships in a broader population.

Which One to Use?

- `KINSHIP`: Ideal for understanding the degree of relatedness. Use this if your goal is to identify familial clusters or population structure.
- `IBS0`: Useful for detecting distant relationships or identifying unrelated pairs in large datasets.

FROH