Chapter 3

Data Management and Summary Statistics with PLINK

Christopher C. Chang

Abstract

PLINK is a versatile program which supports data management, quality control, and common statistical computations on matrices of genomic variant calls, in a computationally efficient manner. In population genomics, it is frequently used to take care of the "basics," so they do not need to be reimplemented when a new type of analysis needs to be performed on such a matrix. I describe several of these basic operations, and discuss uses and pitfalls.

Key words Allele frequency, Hardy–Weinberg equilibrium, Linkage disequilibrium, Principal component analysis, Relationship inference, Sex inference, Variant call format

1 Introduction

Genotyping chips and sequencing machines produce data in a wide variety of formats. However, they are all trying to measure the same thing: what are the genome sequences of these organisms? These sequences will tend to be 99%+ -identical between different organisms of the same species, of course, but thanks to mutation and sexual reproduction, interesting variation will remain, and it is this variation that is the primary object of study for population genomics.

The most commonly studied type of variation is the "single nucleotide polymorphism" (SNP), an isolated position in the genome that noticeably varies between organisms of the same species while adjacent positions remain identical. They account for a large fraction of total variation, they are relatively easy to detect with modern technologies, and they introduce fewer analytical difficulties than, e.g., genomic rearrangements. Thus, many population-genomic datasets are in the form of a SNP × sample matrix. PLINK 1.0 [1] introduced a simple and efficient binary encoding for biallelic-SNP × sample matrices which has become a de facto standard (*see* **Note 1**).

Julien Y. Dutheil (ed.), *Statistical Population Genomics*, Methods in Molecular Biology, vol. 2090,
https://doi.org/10.1007/978-1-0716-0199-0_3,

PLINK itself also supports a variety of common data management and quality control operations on such matrices, along with some useful summary statistics; and the wider ecosystem of software directly supporting the PLINK 1 binary format can handle much more (see, e.g., the ADMIXTURE software discussed in Chapter 4). In this chapter, I will cover the following:

- How to convert data into the PLINK 1 binary format.
- Filtering out samples and SNPs with too much missing data.
- Selecting a sample subset without very close relatives.
- Minor allele frequency reporting, and using MAFs to filter out SNPs.
- Hardy–Weinberg equilibrium statistics, and filtering applications.
- Selecting a SNP subset in approximate linkage equilibrium.
- Principal component analysis.
- Sex validation and imputation.
- Reporting linkage disequilibrium statistics.
- Exporting data to other file formats.

2 Materials

You will want PLINK 1.9 and 2.0 [2] for everything that follows. If they are already installed on your system, typing `plink` or `plink2` with no additional command-line arguments into a `bash` (or Windows `cmd`) shell should cause version information and partial lists of supported commands to be printed:

```
~$ plink
PLINK v1.90b6.4 64-bit (7 Aug 2018)           www.cog-genomics.org/plink/1.9/
(C) 2005-2017 Shaun Purcell, Christopher Chang   GNU General Public License v3

plink [input flag(s)...] {command flag(s)...} {other flag(s)...}
plink --help {flag name(s)...}

Commands include --make-bed, --recode, --flip-scan, --merge-list,
--write-snplist, --list-duplicate-vars, --freqx, --missing, --test-mishap,
--hardy, --mendel, --ibc, --impute-sex, --indep-pairphase, --r2, --show-tags,
--blocks, --distance, --genome, --homozyg, --make-rel, --make-grm-gz,
--rel-cutoff, --cluster, --pca, --neighbour, --ibs-test, --regress-distance,
--model, --bd, --gxe, --logistic, --dosage, --lasso, --test-missing,
--make-perm-pheno, --unrelated-heritability, --tdt, --dfam, --qfam, --tucc,
--annotate, --clump, --gene-report, --meta-analysis, --epistasis,
--fast-epistasis, and --score.

'plink --help | more' describes all functions (warning: long).
```

If either is not already installed on your system, prebuilt binaries for Linux, OS X, and Windows can be downloaded from

https://www.cog-genomics.org/plink/1.9/ and
https://www.cog-genomics.org/plink/2.0/

Alternatively, you can download the source code from

https://github.com/chrchang/plink-ng

and build the programs yourself. On some systems, you can also easily install PLINK 1.9 from a package manager such as APT, Bioconda, or Homebrew.

PLINK is designed to interoperate well with R, and this chapter includes an R plotting command. On the off chance you do not have R installed, it can be downloaded from

https://cran.r-project.org/mirrors.html

Sample datasets (based on 1000 Genomes phase 1 [3]) and PLINK outputs for many operations described in this chapter can be downloaded from the companion website.

3 Methods

3.1 Getting Started: Importing and Merging Data

A PLINK 1 binary fileset contains three files:

- A text file with the ".fam" extension, containing sample IDs and possibly some pedigree information (sex, parental IDs). PLINK sample IDs normally have two components: a family ID ("FID") in the first column and an individual ID ("IID") in the second column of the .fam file.
- A text file with the ".bim" extension, containing biallelic-variant IDs, positions, and the two observed alleles for each variant. (Alleles can contain more than one nucleotide; PLINK is designed to work with SNP-like data, but it is not restricted to just SNPs.) Usually, the less common allele is in the 5th column and the more common one is in the 6th, but if your organism has already been sequenced enough to have an official "reference genome," the 6th column may always contain the reference-genome allele.

  PLINK 1.9 refers to the 5th-column allele as "A1," and the 6th-column allele as "A2."
- A binary file with the ".bed" (*see* **Note 2**) extension, containing a compact representation of the variant × sample matrix. Logically, for an autosome in a diploid genome, each matrix entry is either "0 copies of A2 allele," "1 copy," "2 copies," or "NA."

Sometimes, you will be given data in a different format, and/or multiple filesets which should be merged into a single PLINK binary fileset for more convenient analysis.

3.1.1 Variant Call Format

Genome sequencing data is frequently represented in Variant Call Format (VCF) [4] or its binary counterpart BCF. In addition to variants and genotype calls, these files may contain genotype and mapping quality statistics, and many other tidbits of information. However, in population genomics, you usually only need the variants and genotype calls. These can be converted to PLINK-format with a command like

```
plink --vcf original_data.vcf.gz \
    --keep-allele-order \
    --make-bed \
    --out converted_data
```

which generates a {converted_data.bed, converted_data.bim, converted_data.fam} fileset with the same genotype calls as original_data.vcf.gz.

Let us walk through the pieces of the command line above.

- "–vcf original_data.vcf.gz" specifies that the primary source of input data for this run is original_data.vcf.gz, and it is in VCF format. The .gz at the end of the filename indicates that the file is gzip-compressed; PLINK automatically decompresses the file in this case.
- The backslash at the end of the first line indicates that we are not done typing the command. It is not strictly necessary: you could put all four flags on a single line instead. But the one-flag-per-line style is usually more readable.
- "–keep-allele-order" tells PLINK 1 to keep the allele ordering in the input file, because VCF files explicitly specify which allele is in the species "reference genome" and which allele deviates from it, and that information is useful at times. A2 is set to the reference allele, and A1 is set to the alternate allele.

  Without this flag, PLINK 1 automatically reorders the alleles such that A2 is the most common ("major") allele *in the immediate dataset*, and A1 is the least common ("minor"). Thus, it is normally unsafe to make assumptions about which allele is A1 and which is A2: if a particular allele is present at 48% frequency, it is easy to imagine that allele having 51% frequency in one dataset (and thus being "major" there) and 45% in another. However, there are several ways to impose a specific ordering when necessary; –ref-from-fa (PLINK 2 only) and –a2-allele are generally the most useful.
- "–make-bed" is the command to generate a new PLINK 1 binary fileset. This is used a lot.
- "–out" lets you set the output filename prefix for the current run. Without it, all output filenames would be of the form plink.{extension} instead.

You can find more documentation on each of these flags by going to

https://www.cog-genomics.org/plink/1.9/

and typing the flag name into the "Quick index search" box at the bottom of the left sidebar. Or you can run "plink –help [flag name]" to get a quick summary. Both sources of documentation also discuss some quality filters PLINK can apply, if very-low-quality genotype calls remain in the VCF file; the information on how PLINK sample IDs are generated from VCF sample IDs may also be relevant.

The command line

```
plink2 --vcf original_data.vcf.gz \
     --make-bed \
     --out converted_data
```

has the same effect. Note the lack of "–keep-allele-order": PLINK 2's default assumption is that the 6th .bim column contains reference-genome alleles, and you need to add "–maj-ref" to tell PLINK 2 to reorder the alleles the same way PLINK 1 does. This reflects the fact that reference genomes are more widely agreed on and more stable than they were in 2007.

For brevity, we will ignore –keep-allele-order, –a2-allele, and the like in the rest of this chapter, but be aware that allele order matters sometimes. We will also just give the PLINK 1.9 form of a command when either PLINK version can be used.

3.1.2 PLINK text ({.ped, .map})

A significant number of older datasets are in PLINK's original text fileset format, where the .map file contains variant IDs and positions, and the .ped file stores both sample IDs/pedigree info and genotype calls. These can be imported with

```
plink --file original_data \
    --make-bed \
    --out converted_data
```

Replace –file with –tfile to import a "transposed text" fileset ({.tped, .tfam}).

3.1.3 Other Formats

PLINK is capable of importing several other commonly used genotype data formats. The necessary command lines are all very similar; you usually just need to replace –vcf/–file with another flag. Refer to

https://www.cog-genomics.org/plink/1.9/input and
https://www.cog-genomics.org/plink/2.0/input

for details.

3.1.4 Alternate Chromosome/Contig Sets

Unless you specify otherwise, PLINK interprets chromosome codes as if you were working with human data: 1–22 (or "chr1"–"chr22") refer to the respective autosomes, 23 refers to the X chromosome, 24 refers to the Y chromosome, 25 refers to pseudoautosomal regions, and 26 refers to mitochondria.

If you are working with another species, use the "–chr-set" flag to indicate the number of autosomes. A positive parameter indicates a diploid genome, while a negative number tells PLINK to treat the genome as haploid; so "–chr-set 38" is appropriate for dog data, while "–chr-set -19" is correct for some bog moss species.

Some datasets also contain unplaced contigs. As long as their names start with non-numeric characters (e.g., "contig35" is ok, "35" is not), PLINK will accept them when you include "–allow-extra-chr" (–aec for short) on the command line.

(Unfortunately, if you need –chr-set or –aec once, you will usually need to include it *every* time you run PLINK on data for that species.)

3.1.5 Missing Variant IDs

Sometimes, your input files contain lots of variants which have not been assigned a unique ID (e.g., a VCF file where lots of "ID" column entries are "."). This can prevent some PLINK commands from working properly (such as the fileset merge operation we will discuss next), so it is best to address this during or immediately after data import.

PLINK's –set-missing-var-ids flag provides one solution. Given a template string where "@" represents the chromosome code and "#" represents the base-pair coordinate on the chromosome, it replaces every "." variant ID with a template-based ID. For example,

```
plink --bfile converted_data \
    --set-missing-var-ids @:# \
    --make-bed \
    --out idfilled_data
```

would assign the ID "3:5331691" to an unnamed variant at chromosome 3, position 5331691. This is good enough for SNP-only data.

PLINK 2 also allows the template string to include REF/ALT allele codes, supports several ways of handling very long allele codes, and lets you rewrite all variant IDs with the template instead of just missing IDs; see https://www.cog-genomics.org/plink/2.0/data#set_all_var_ids.

3.1.6 Merging

So you have converted all your data to PLINK 1 binary format, but it is split across multiple filesets. To merge them into a single fileset,

- Create a text file with each of the input filename prefixes, one per line. Call this input_sources.txt.

- Run

```
plink --merge-list input_sources.txt \
    --out merged_data
```

Note that, if you are merging a thousand single-sample input filesets which all have the same sample ID, PLINK will assume all genotype calls are for the same individual (and, as a consequence, most or all genotype calls in the merged dataset will be missing; PLINK's merger normally only keeps a genotype call when all input files agree on it). Assign a different sample ID to each individual before merging.

3.1.7 Filling in Missing Pedigree Information

VCF files normally do not contain pedigree information. However, you will often know at least the sexes of most of the samples anyway. Here is one way to integrate them with your merged PLINK fileset.

- Create a text file where the first two columns are PLINK sample IDs, and the third column indicates sex (1 or M = male, 2 or F = female) (*see* **Note 3**). Call this file sex_info.txt.
- Run

```
plink --bfile merged_data \
    --update-sex sex_info.txt \
    --make-bed \
    --out merged_data_with_sex
```

You can import parental IDs in a similar manner; see the documentation on –update-parents. (Or you can write your own script to manipulate the .fam file; there is more than one way to do these things.)

3.2 Missingness Filters

As of this writing, genotyping chips and genome sequencers are not perfect. For this and other reasons, many datasets contain a bunch of "NA" entries in the variant × sample genotype call matrix.

The usual practice is to filter out the samples and variants with high missing-entry frequencies; these tend to be caused by mistakes in the lab, bad SNP probes, variant calling limitations, and similar issues where throwing out the entire row/column is an appropriate solution. (You still have lots of other rows and columns to work with, so at least in population genomics it usually is not worth the effort to try to salvage it.) PLINK's –mind (for **ind**ividual **m**issingness) and –geno flags provide a way to do this:

```
plink --bfile merged_data \
    --geno 0.1 \
    --mind 0.1 \
    --make-bed \
    --out missingness_filtered_data
```

Walking through this command line,

- –bfile specifies that the primary source of input data for this run is {merged_data.bed, merged_data.bim, merged_data.fam}.
- "–geno 0.1" tells PLINK to throw out every variant where more than 10% of the genotype calls are "NA"s.
- "–mind 0.1" tells PLINK to throw out every sample where more than 10% of the genotype calls are "NA"s. This happens *before* –geno, not simultaneously with it (or after it; command-line flag order is ignored). You can see PLINK 1.9's full order of operations at

  https://www.cog-genomics.org/plink/1.9/order
- –make-bed generates a new PLINK 1 fileset, with the high-missingness samples and variants removed. When using PLINK, you generally do not "edit" data in-place; instead you generate a new fileset with a different name whenever you apply filters or other data transformations.
- –out causes the new files to be written to {missingness_filtered_data.bed, missingness_filtered_data.bim, missingness_filtered_data.fam} instead of {plink.bed, plink.bim, plink.fam}.

  Most PLINK runs also generate a .log file which includes the original command line as well as other information printed to the console; in this case, it would be named missingness_filtered_data.log. As long as you do not delete these .log files or reuse the same –out arguments at inappropriate times, this makes it easy to see how each PLINK output file in a directory was generated.

3.3 Selecting a Sample Subset Without Very Close Relatives

Many population-genomic statistics (such as the allele frequencies we are about to discuss) and analyses are distorted when there are lots of very close relatives in the dataset; you are generally trying to make inferences about the population as a whole, rather than a few families that you oversampled. PLINK 2 includes an implementation of the KING-robust [5] pairwise relatedness estimator, which can be used to prune all related pairs (*see* **Note 4**). For example, to get rid of all first-degree relations (parent–child and sibling–sibling), you could run

```
plink2 --bfile missingness_filtered_data \
       --king-cutoff 0.177 \
       --make-bed \
       --out relpruned_data
```

If you want to do fancier things based on this relatedness estimator, such as trying to reconstruct the entire pedigree, take a look at the KING program, which can be downloaded from http://people.virginia.edu/~wc9c/KING/Download.htm

3.4 Minor Allele Frequency Reporting, Filtering

Allele frequencies are a primary object of study in population genetics and genomics. PLINK's —freq command reports empirical allele frequencies, and its —maf filter removes all variants with minor allele frequency below the given threshold.

For example,

```
plink --bfile relpruned_data \
      --freq \
      --out allele_freqs
```

writes an empirical allele frequency report to allele_freqs.frq, where the first column contains the chromosome, the second column contains the variant ID, columns 3–4 contain the minor and major allele codes in that order, column 5 contains the minor allele frequency, and column 6 contains the number of allele observations. (This file format is spelled out at

https://www.cog-genomics.org/plink/1.9/formats#frq,

and all other PLINK 1.9 output file formats are described elsewhere on the page.)

If you are interested in the allele frequency spectrum (refer to Chapter 1 for some of its applications), PLINK 2 —freq has an additional convenience that may come in handy:

```
plink2 --bfile relpruned_data \
     --freq alt1bins=0.01,0.02,0.03,0.04,0.05,0.1,0.2,0.3,0.4 \
     --out allele_spectrum
```

This generates an additional allele_spectrum.afreq.alt1.bins file which reports the number of variants with MAF in $[0, 0.01)$, $[0.01, 0.02)$, …, and $[0.4, 0.5]$. (And the main allele frequency file is formatted slightly differently from PLINK 1.9, with a more VCF-like header row, and single-tab delimiters instead of multiple spaces. These changes are small enough that it should not be difficult to switch an old codebase from PLINK 1 to PLINK 2, but be aware that the latter is not a drop-in replacement for the former.)

To filter out all variants with minor allele frequency below 5%, you would run

```
plink --bfile relpruned_data \
    --maf 0.05 \
    --make-bed \
    --out maf_filtered_data
```

3.5 Hardy–Weinberg Equilibrium Statistics

Checking for deviation from Hardy–Weinberg equilibrium is useful for multiple reasons:

- If there are far more heterozygous calls than would be expected under Hardy–Weinberg equilibrium, that is usually due to a systematic variant calling error. Any such variants should be removed from the dataset.
- Population stratification can cause large violations of Hardy–Weinberg equilibrium, in the fewer-hets-than-expected direction. Natural selection, migration, and some mating patterns can drive smaller violations in either direction.
- Several statistical methods assume approximate Hardy–Weinberg equilibrium. You can use a more-stringently-filtered subset of your data for just these methods.

PLINK includes functions for computing Hardy–Weinberg equilibrium exact test p-values, based on the method of Wigginton et al. [6] The –hwe flag filters out variants with p-values more extreme than a given threshold; the following PLINK 2 command line is appropriate during initial quality control:

```
plink2 --bfile maf_filtered_data \
     --hwe 1e-25 keep-fewhet \
     --make-bed \
     --out hwe_filtered_data
```

The "keep-fewhet" modifier causes this filter to be applied in a one-sided manner (so the fewer-hets-than-expected variants that one would expect from population stratification would *not* be filtered out by this command), and the 1e-25 threshold is extreme enough that we are unlikely to remove anything legitimate. (Unless the dataset is primarily composed of F1 hybrids from an artificial breeding program.) After you have a good idea of population structure in your dataset, you may want to follow up with a round of two-sided –hwe filtering, since large (*see* **Note 5**) violations of Hardy–Weinberg equilibrium in the fewer-hets-than-expected direction *within* a subpopulation are also likely to be variant calling errors; with multiple subpopulations, the –write-snplist and –extract flags can help you keep just the SNPs which pass all subpopulation HWE filters.

The –hardy command can be used to dump all of the p-values.

3.6 Selecting a SNP Subset in Approximate Linkage Equilibrium

Some analyses, such as the PCA we will discuss next, work best on a genome-spanning subset of SNPs which are in approximate linkage equilibrium. PLINK 2's –indep-pairwise command (*see* **Note 6**) is a computationally efficient method of identifying a reasonable subset. For example,

```
plink2 --bfile hwe_filtered_data \
     --indep-pairwise 200kb 1 0.5 \
     --out ldpruned_snplist
```

removes SNPs so that no pair within 200 kilobases have squared-allele-count-correlation (r^2) greater than 0.5, and saves the IDs of the remaining SNPs to ldpruned_snplist.prune.in. (There is nothing magical about the $r^2 = 0.5$ threshold; it is useful to adjust it depending on the number of SNPs you want to keep. The lower the threshold you use, the larger your kilobase window should be.)

You can then create a fileset with just this SNP subset by combining –extract (which lets you select an arbitrary subset of variants, by ID) with –make-bed:

```
plink --bfile hwe_filtered_data \
    --extract ldpruned_snplist.prune.in \
    --make-bed \
    --out ldpruned_data
```

3.7 Principal Component Analysis

Once you have LD-pruned and MAF-filtered your dataset, PLINK 2's –pca command has a good shot of revealing large-scale population structure. For example,

```
plink2 --bfile ldpruned_data \
     --pca 5 \
     --out pca_results
```

writes a tab-delimited table to pca_results.eigenvec, with one sample per row and one principal component per later column. (Eigenvalues are written to pca_results.eigenval.)

You will usually want to sanity-check the output at this point, and verify that the top principal components do not correlate too strongly with, e.g., sequencing facility or date. (A full discussion of "batch effects" and how to deal with them could take up an entire chapter; worst case, you may have to analyze your batches separately, or even redo all genotyping/sequencing from scratch. I will be optimistic here and suppose that no major problem was uncovered by PCA, but be aware that this is frequently your best chance to catch data problems that would otherwise sink your entire analysis.)

It is also a good idea to throw out gross outliers at this point; any sample which is more than, say, 8 standard deviations out on any top principal component is likely to have been genotyped/sequenced improperly; you can remove such samples by creating a text file with the bad sample IDs, and then using –remove + –make-bed:

```
plink --bfile ldpruned_data \
    --remove bad_samples.txt \
    --make-bed \
    --out ldpruned_data2
```

If you do this, follow it up by repeating the PCA, since the bad samples might have distorted the principal components:

```
plink2 --bfile ldpruned_data2 \
     --pca 5 \
     --out pca_results2
```

(Occasionally, the new principal components will reveal *another* bad sample, and you have to repeat these two steps, etc. EIGENSOFT [7, 8] has some additional built-in principal component analysis options, including automated iterated outlier removal, and a top-eigenvalue-based test for significant population structure.)

Anyway, once there is nothing obviously wrong with the PCA results, you can load the table in R and plot the top pairs of principal components against each other:

```
> pca_table <- read.table("pca_results.eigenvec", header=TRUE, comment.char="")
> plot(pca_table[, c("PC1", "PC2", "PC3", "PC4", "PC5")])
```

Results are shown in Fig. 1.

If there are obvious clusters in the first few plots, I recommend jumping ahead to Chapter 4 (on ADMIXTURE) and using it to label major subpopulations before proceeding.

3.8 Sex Validation and Imputation

If you have X-chromosome population-genomic data, you can employ PLINK's –check-sex command to sanity-check the sex information in your .fam file. (The method is based on chrX heterozygosity rates.) Similarly, –impute-sex uses chrX heterozygosity rates to fill in missing sex entries when appropriate.

This is typically a two- or three-step process:

0. If your species has pseudoautosomal regions, ensure they are not encoded as part of the X chromosome. If they are, PLINK 1.9's –split-x or PLINK 2's –split-par flag can be used to change the chromosome codes of the relevant variants before proceeding.
1. Run –check-sex once without additional parameters, just to see the distribution of F (inbreeding) coefficients.

```
plink --bfile ldpruned_data \
    --check-sex \
    --out f_distribution
```

Then plot the distribution of values in the sixth column of f_distribution.sexcheck. If both genders are well-represented in the dataset, you should see a big tight clump near 1 (corresponding to the males), and a more widely dispersed set of values centered near 0 (corresponding to the females).

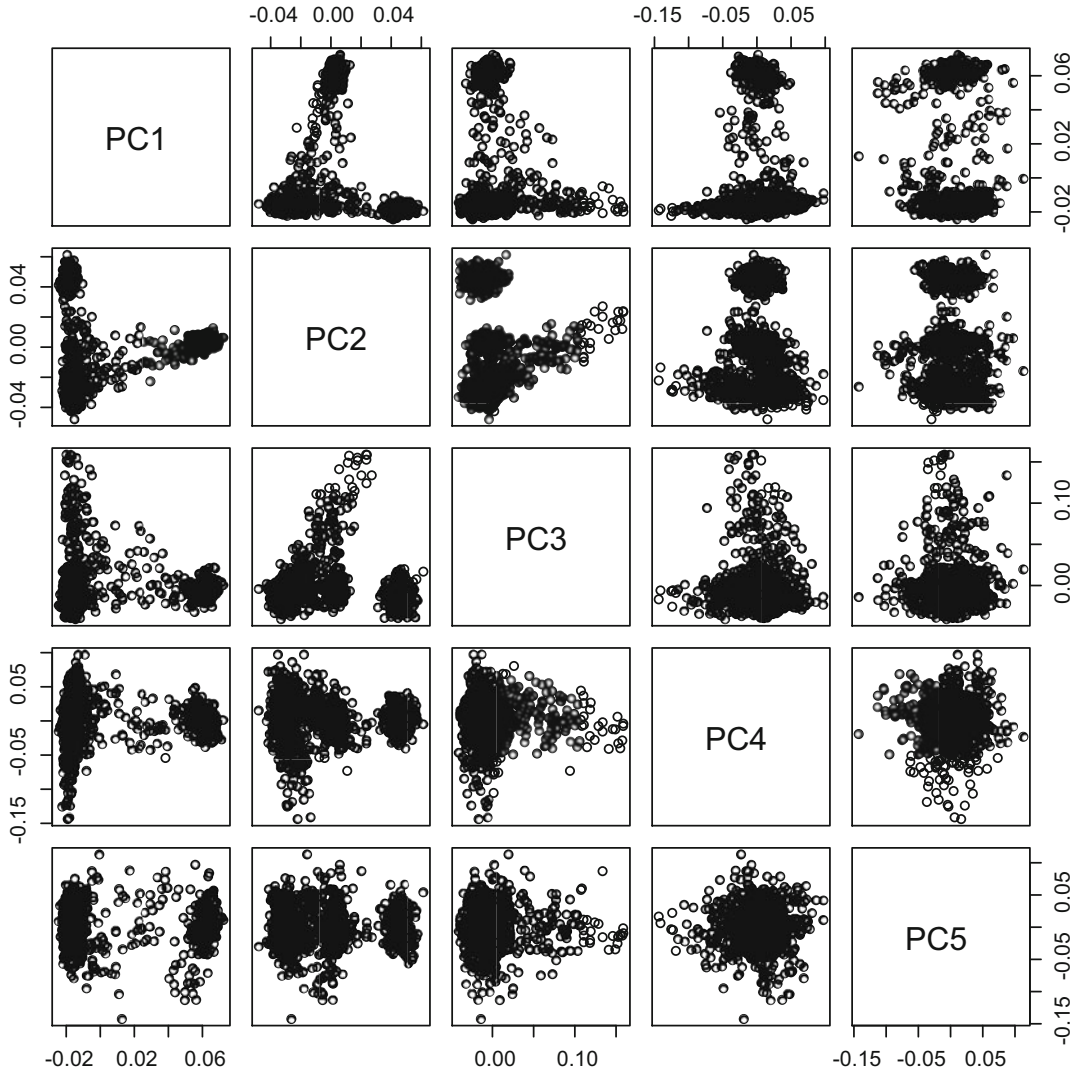

Fig. 1 PCA bi-plot generated by R

2. Select a pair of decision boundaries (where an F coefficient below the first boundary causes the sample to be classified as female, above the second boundary causes the sample to be classified as male, and in between causes the sample to be labeled as unknown-sex), and run –impute-sex (or just –check-sex again) with the boundaries:

```
plink --bfile ldpruned_data \
    --impute-sex 0.7 0.8 \
    --make-bed \
    --out sexfilled_data
```

If you also have Y-chromosome genotype calls and can expect males to have fewer missing chrY calls than females, you can take this into account in the sex inference process; refer to the –check-sex/–impute-sex documentation at

https://www.cog-genomics.org/plink/1.9/basic_stats#check_sex

3.8.1 Subpopulation Allele Frequencies, and –read-freq

By default, –check-sex/–impute-sex and several other PLINK commands assume that your dataset is representative of a single population, and all samples are members of that population; population allele frequencies and F coefficients are estimated under these assumptions.

Thus, if you identified multiple subpopulations in the previous section, you should perform sex validation/imputation on one subpopulation at a time. PLINK's –filter flag provides one way to do this; put sample IDs in the first two columns of subpops.txt and subpopulation IDs in the third column, then

```
plink --bfile ldpruned_data \
    --filter subpops.txt AFR \
    --make-bed \
    --out afr_data

plink --bfile afr_data \
    --check-sex \
      ...
```

You can then use –update-sex to copy the inferred sexes back to your main datasets.

Also, you sometimes have access to more accurate (sub)population allele frequencies than would be imputed from your immediate dataset. (An extreme case of this is when you are running –check-sex/–impute-sex separately for a bunch of single-sample filesets.) To patch in the more-accurate allele frequencies, reformat them as a PLINK –freq report if necessary, and then use –read-freq:

```
plink --bfile ldpruned_data \
    --read-freq more_accurate.frq \
    --check-sex \
    --out f_distribution
```

3.9 Reporting Linkage Disequilibrium Statistics

While the last several operations worked with a LD-pruned subset of the data, sometimes you will want to handle linkage disequilibrium in a more sophisticated manner, or study it directly. PLINK's –r2 command can be used for this purpose.

```
plink --bfile hwe_filtered_data \
    --r2 dprime \
    --ld-window 999999 \
    --ld-window-kb 1000 \
    --ld-window-r2 0.1 \
    --out r2_report
```

The command line above reports squared-allele-count-correlations and an estimate of Lewontin's D' (thanks to the additional "dprime" modifier) for each pair of variants within 1000 kilobases of each other, whenever $r^2 \geq 0.1$. The "–ld-window 999999" flag is needed because PLINK defaults to only considering variant pairs which are at most 9 lines apart from each other in the .bim file.

–r2 has a bunch of other options, including matrix output (when you want to look at literally every single pair of variants) and ways to focus on just a few SNPs; see the documentation at

https://www.cog-genomics.org/plink/1.9/ld#r

3.10 Data Export

While the PLINK 1 binary fileset format is widely supported, occasionally you will need to convert to a different format like VCF or PLINK text in order to use another tool. This can be accomplished with the PLINK –export command:

```
plink --bfile hwe_filtered_data \
    --export ped \
    --out exported_plink_text

plink2 --bfile hwe_filtered_data \
     --ref-from-fa reference.fa \
     --export vcf \
     --out exported_vcf
```

The first command generates {exported_plink_text.ped, exported_plink_text.map}, while the second one generates exported_vcf.vcf, determining the correct REF alleles from reference.fa in case they were scrambled by PLINK 1.9. Many other formats are supported; see

https://www.cog-genomics.org/plink/1.9/data#recode and
https://www.cog-genomics.org/plink/2.0/data#export.

4 Notes

1. SNPs with three or four alleles exist, but are rare enough such that most software developers ignore them. The usual practice as of this writing is to "split" a triallelic SNP into two biallelic records; for example, if A, C, and G nucleotides are all observed

at a single position and A is the most common, the dataset will contain one record describing an A/C SNP, and another record describing an A/G SNP. This is obviously imperfect, but it is good enough for most genomic analyses.

2. Unfortunately, there is another type of ".bed" file widely used in genomics: UCSC Browser Extensible Data, a text format for positional intervals. However, you can usually distinguish between the two simply by checking for the presence of .bim and .fam files with the same filename prefix; these are practically required to make sense of a PLINK .bed file. Or, in a pinch, you can check whether the first three bytes of the .bed file are consistent with the specification at https://www.cog-genomics.org/plink/1.9/formats#bed.
3. Assuming your species adheres to the XY sex-determination system. For ZW species, you may want to deliberately reverse the sexes, and encode Z→X and W→Y.
4. This does not mean that both samples in each related pair are thrown out. Instead, –king-cutoff tries to keep as much data as possible, and as a consequence it usually keeps one sample out of each pair.
5. What p-value threshold does "large" correspond to, you may ask? Well, this depends on the size of your dataset and some other characteristics of your data. But a good rule of thumb is to use "ridiculous" thresholds like 1e-25 or 1e-50 for quality control when you have at least a thousand samples (or even 1e-200 if you have many more), while only using "normal" thresholds like 1e-4 or 1e-7 when you are preparing data for an analysis which actually assumes all your SNPs are in Hardy–Weinberg equilibrium.
6. The –indep-pairwise command also works in PLINK 1.9, but it may run a lot more slowly since it is not automatically parallelized.

References

1. Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira M, Bender D, Maller J, Sklar P, de Bakker P, Daly M, Sham P (2007) PLINK: a tool set for whole-genome association and population-based linkage analyses. Am J Hum Genet 81:559–575
2. Chang C, Chow C, Tellier L, Vattikuti S, Purcell S, Lee J (2015) Second-generation plink: rising to the challenge of larger and richer datasets. GigaScience 4:7
3. The 1000 Genomes Project Consortium (2012) An integrated map of genetic variation from 1,092 human genomes. Nature 491:56–65
4. Danecek P, Auton A, Abecasis G, Albers C, Banks E, DePristo M, Handsaker R, Lunter G, Marth G, Sherry S, McVean G, Durbin R (2011) The variant call format and vcftools. Bioinformatics 27:2156–2158
5. Manichaikul A, Mychaleckyj J, Rich S, Daly K, Sale M, Chen W (2010) Robust relationship

inference in genome-wide association studies. Bioinformatics 26:2867–2873
6. Wigginton J, Cutler D, Abecasis G (2005) A note on exact tests of Hardy-Weinberg equilibrium. Am J Hum Genet 76:887–893
7. Patterson N, Price A, Reich D (2006) Population structure and eigenanalysis. PLoS Genet 2
8. Price A, Patterson N, Plenge R, Weinblatt M, Shadick N, Reich D (2006) Principal components analysis corrects for stratification in genome-wide association studies. Nat Genet 38:904–909