Genotyping chip data lift-over to reference genome build GRCh38/hg38

Kalle Pärn,Javier Nunez Fontarnau,Marita A. Isokallio,Timo Sipilä,Elina Kilpelainen,Aarno Palotie,Samuli Ripatti,Priit Palta

Abstract

Citation: Kalle Pärn,Javier Nunez Fontarnau,Marita A. Isokallio,Timo Sipilä,Elina Kilpelainen,Aarno Palotie,Samuli Ripatti,Priit Palta Genotyping chip data lift-over to reference genome build GRCh38/hg38. **protocols.io**
dx.doi.org/10.17504/protocols.io.nqtddwn
Published: 10 Apr 2018

Protocol

Requirements and preparatory steps

Step 1.

The protocol is aimed for lifting genotyping chip data from older reference genome build versions over to human reference genome build version GRCh38/hg38.

For a 'quick and dirty' lift-over you can jump straight to **Step 2** and only run that.

For high-quality and verified results we suggest running through **Steps 1-9**.

If you additionally need to convert your chip data from PLINK format to VCF format, also follow **Steps 10-11**.

Throughout the protocol we assume Bash shell.

This **Step 1** defines the requirements for the protocol (e.g. required software packages and reference files) and suggests example commands how to process the files into suitable formats.

1.1 Software packages

1.1.1 Download and install the software packages

Required software packages are listed below with the versions that were used in the protocol below. However, use of the newest versions is recommended.

- PLINK v1.9 http://www.cog-genomics.org/plink/1.9/
- R v3.3.0 https://www.r-project.org/
- R package data.table https://github.com/Rdatatable/data.table/wiki/Installation

1.1.2 Optional software packages (required for VCF conversion Steps 10 and 11)

- BCFtools v1.7 http://www.htslib.org/download/
- bgzip is part of the BCFtools package

1.1.3 Export the paths

Once installed, export the correct paths to environmental variable PATH:

```
echo
PATH=$PATH:/path/to/installed/plink/executable/dir/:/path/to/installed/R/exec
utable/dir/:/path/to/installed/bcftools/executable/dir/ >> $HOME/.bashrc

source $HOME/.bashrc
```

1.1.4 Install the R package

Once R is installed, the 'data.table' package can be installed **in R**, e.g.:

```
install.packages('data.table', type = 'source', repos =
'http://Rdatatable.github.io/data.table')
```

or, alternatively, if you already downloaded the package:

```
install.packages('/path/to/the/downloaded/data.table_1.10.4.tar.gz', repos =
NULL, type = 'source')
```

1.2. Reference data (variant allele frequency file)

Reference variant allele frequency file is required for comparison of the chip genotyped variant allele frequencies in order to identify possible allele swaps and variants with unexpected allele frequency discrepancies.

Note: Chromosome notation in the reference data should follow the GRCh38/hg38 notations ('chr#' for autosomal chromosomes and 'chrX' for chromosome 23).

1.2.1 Obtain the reference variant allele frequency data

If population-specific data are available (e.g. from a corresponding WGS effort) using these data would be preferrable.

Process the data as instructed in **Steps 1.2.2-1.2.4**.

If population-specific reference data is not available, for instance 1000 Genomes Project (www.nature.com/articles/nature15393) data can be used instead.

We have prepared (as described below in **Steps 1.2.2-1.2.4**) the 1000 Genomes Project GRCh38/hg38 data (downloaded from the EBI FTP site: ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/supporting/GRCh38_positions/) to generate variant allele frequency files for 1000GP ALL and EUR samples.

The corresponding allele frequency files are available for downloading at Google Cloud: https://console.cloud.google.com/storage/browser/fimm-public-data/1000GP_frq_files/

If you use these files, you can skip creating the frequency file and **jump to Step 1.3**. Otherwise (if you want to create your own/alternative 1000GP-based custom reference frequency file), follow the steps below.

1.2.2 Check for multiallelic sites

Confirm that multiallelic sites (if present) in your reference data files are decomposed. If they are not, use the example command below to split the multiallelic sites into biallelic records:

```
for chr in {1..23}; do
    bcftools norm -m -any ref_data_chr${chr}.vcf.gz -Oz -o
ref_data_split_multiallelic_chr${chr}.vcf.gz
done
```

1.2.3 Check the chromosome notation

Confirm that the chromosome notation in your reference data files follows the GRCh38/h38 notations as '**chr#**' for autosomal, '**chrX**' for chromosome 23 and '**chrM**' for mitochondrial sites.

If not, see **Step 11** for an example command to rename the chromosomes before proceeding.

1.2.4 Generate the allele frequency file

Generate a tab-delimited file of the reference data allele frequencies, one line per variant, with columns CHR, SNP (as CHR_POS_REF_ALT), REF, ALT, AF (including the header line).

Use the reference data VCF files as input with the example command below and save the generated frequency file as:

- ref_data.frq

```
# Check your reference data VCF and if it does NOT contain AF in the INFO
field, calculate it with BCFTools +fill-tags plugin
# Note: BCFTools plugins require environmental variable BCFTOOLS_PLUGINS
exported
export BCFTOOLS_PLUGINS=/path/to/bcftools/plugins_folder
# Calculate AF for each chromosome VCF file
for chr in {1..23}; do
    bcftools +fill-tags ref_data_chr${chr}.vcf.gz -Oz -o
ref_data_AF_chr${chr}.vcf.gz -- -t AF
done

# Generate a tab-delimited header for the allele frequency file
echo -e 'CHR\tSNP\tREF\tALT\tAF' > ref_data.frq

# Query the required fields from the reference VCF files and append to the
allele frequency file
for chr in {1..23}; do
    bcftools query -f
'%CHROM\t%CHROM\_%POS\_%REF\_%ALT\t%REF\t%ALT\t%INFO/AF\n'
ref_data_chr${chr}.vcf.gz >> ref_data.frq
done
```

Note: Chromosome notation should follow the GRCh38/hg38 notations ('chr#' for autosomal chromosomes and 'chrX' for chromosome 23).

1.3. You are ready to start!

As the last prepatory step, let's go over the required input data file(s) and also expected final output files!

1.3.1 Protocol input files

Input data is the chip genotype data in PLINK format (here v1.9).

More details on the PLINK formats can be found at PLINK web page:
http://www.cog-genomics.org/plink/1.9/formats

- **<dataset>.bed** - binary representation of genotype calls

- **<dataset>.bim** - extended variant information file including six columns:

1. Chromosome code (either an integer, or 'X'/'Y'/'XY'/'MT'; '0' indicates unknown) or name
2. Variant identifier
3. Position in morgans or centimorgans (safe to use dummy value of '0')
4. Chromosomal base-pair coordinate
5. Allele 1 (corresponding to clear bits in .bed; usually minor)
6. Allele 2 (corresponding to set bits in .bed; usually major)

- **<dataset>.fam** - sample information file including six columns:

1. Family ID ('FID')
2. Within-family ID ('IID'; cannot be '0')
3. Within-family ID of father ('0' if father isn't in dataset)
4. Within-family ID of mother ('0' if mother isn't in dataset)
5. Sex code ('1' = male, '2' = female, '0' = unknown)
6. Phenotype value ('1' = control, '2' = case, '-9'/'0'/non-numeric = missing data if case/control)

1.3.2 Final protocol output files:

The final file set are the PLINK format files lifted over to human genome reference build 38 (b38).

Depending on the dataset in question, different data consistency verification steps (**Steps 7-9**) may have been required. Thus, the final output filename tags (<tags>) can vary from run to run.

- <dataset>_b38_<tags>.bed

- <dataset>_b38_<tags>.bim
- <dataset>_b38_<tags>.fam

The final output file from optional **Steps 10** and **11**:

- <dataset>_b38.vcf.gz

Genome build lift-over

Step 2.

If the genotyping chip used an older reference genome version (e.g. GRCh37/hg19) and PLINK format, the data have to be lifted over to human genome build version 38 (GRCh38/hg38).

For the genome build lift-over we suggest Will Rayner's method.

Download your genotyping chip -specific build 38 zip file (strand and position files) and 'update_build.sh' script from: http://www.well.ox.ac.uk/wrayner/strand/

For example:

```
wget http://www.well.ox.ac.uk/wrayner/strand/update_build.sh .
wget
http://www.well.ox.ac.uk/wrayner/strand/your_chip_platform_specific_chip_stra
ndfile-b38-strand.zip .
```

IMPORTANT NOTES:

• 'update_build.sh' uses hardcoded command 'plink', make sure you have exported the path correctly or replace the plink commands with full path to your PLNK installation

• The method assumes the chip data in Illumina TOP strand format. If this is not the case, the variants are incorrectly flipped and need to be back-flipped (see later steps for further instructions).

Uncompress the downloaded files and use the .strand file to run the script with the command below.

Input files:

• <dataset> input file prefix of PLINK format files (.bed, .bim, .fam)
• <chip_strandfiles>.zip strand files corresponding the genotyping chip

Output files:

• <dataset>_b38.bed

• <dataset>_b38.bim

• <dataset>_b38.fam

```
cmd COMMAND
# Uncompress the files
unzip <chip_strandfiles>.zip

# Run the script
./update_build.sh <dataset> <chip_strandfile>.strand <dataset>_b38
```

Lift-over verification

Step 3.

To confirm the successful genome build lift-over, chip data allele frequencies are compared against the reference data allele frequencies. In this step, chip data allele frequency file is generated.

The allele frequency report format is as defined at PLINK site http://www.cog-genomics.org/plink/1.9/formats#frq and the columns are also described below.

1. CHR - Chromosome code (only chromosome number, e.g. '1')
2. SNP - Variant identifier
3. A1 - Usually minor allele
4. A2 - Usually major allele
5. MAF - A1 frequency (values from 0 to 0.5)
6. NCHROBS - Number of allele observations

The PLINK format frequency file needs to be reformatted such that it corresponds to the VCF-style format for GRCh38/hg38:

1. CHR - Chromosome code (with 'chr' tag, e.g. 'chr1')
2. SNP - Variant identifier in format CHR_POS_REF_ALT
3. REF - Reference allele
4. ALT - Alternative allele
5. AF - Allele frequency (values from 0 to 1)

Utilize **.bim** file to first generate the CHR, SNP, REF and ALT columns using A2 as the REF allele, and take AF from the **.frq** file.

Then do the same by using A1 as REF allele and invert the AF by subtracting it from 1. Combine the generated files into a single file representing VCF-style format.

Input files:

• <dataset>_b38.bed

• <dataset>_b38.bim

• <dataset>_b38.fam

Output files:

• <dataset>_vcf_format.frq

cmd COMMAND

```
# First generate a frequency file from the chip data
plink --bfile <dataset>_b38 --freq --out <dataset>_plink

# Remove the PLINK format frequency file header
sed -i -e '1d' <dataset>_plink.frq

# Generate a tab-separated frequency file for the major allele
# Include 'chr' tags to the chromosome names in the first 2 columns as this is the notation
 in hg38
# Add also a SNP column (in format CHR_POS_REF_ALT here columns $1, $4, $6 and $5 of the .b
im file) and a header which match to the header format in the panel.frq file
paste <(awk -
v OFS='\t' '{print "chr"$1, "chr"$1"_"$4"_"$6"_"$5, $6, $5}' <dataset>_b38.bim) <(awk -
v OFS='\t' '{print $5}' <dataset>_plink.frq) | \
awk 'BEGIN{printf "CHR\tSNP\tREF\tALT\tAF\n"} {print $0}' > <dataset>_major_allele.frq

# Generate a tab-separated frequency file for the minor allele
# Include 'chr' tags to the chromosome names in the first 2 columns as this is the notation
 in hg38
# Add also a SNP column (in format CHR_POS_REF_ALT here columns $1, $4, $5 and $6 of the .b
im file) which match to the format in the panel.frq file
paste <(awk -
v OFS='\t' '{print "chr"$1, "chr"$1"_"$4"_"$5"_"$6, $5, $6}' <dataset>_b38.bim) <(awk -
v OFS='\t' '{print 1-$5}' <dataset>_plink.frq) > <dataset>_minor_allele.frq

# Concatenate the two files into a single .frq file
cat <dataset>_major_allele.frq <dataset>_minor_allele.frq > <dataset>_vcf_format.frq

# Ensure that chrX uses the GRCh38/hg38 notation
sed -i 's/chr23/chrX/g' <dataset>_vcf_format.frq
```

Lift-over verification

Step 4.

Compare the chip data allele frequencies (created in **Step 3**) to the reference data allele frequencies (created in **Step 1.2**).

Copy and save the R script below as 'compare_AF.R' and run it as suggested in the command below.

The R script requires data.table package installed (for suggested instructions, see **Step 1.1**).

Inside the R script, intersection of the reference data and chip data matching variants (same SNP ID in format CHR_POS_REF_ALT) are formed and corresponding allele frequencies are plotted against each other.

Input files:

• <dataset>_vcf_format.frq

• ref_data.frq

Output file:

• <dataset>_AF.jpg

Inspect the plot:

See example plots at section 'Expected results' below.

Successful genome build lift-over:

- Nearly all chip data variant allele frequencies correlate with the panel variant allele frequencies and data shows tight, uniform diagonal line.

- Continue with **Step 6**

Unsuccessful genome build lift-over:

- Some of the chip data variant allele frequencies correlate with the reference data variant allele frequencies, whereas some of the chip data show negative correlation with the reference data variant allele frequencies. Hence, an x-shaped plot is observed

- Continue with **Step 5**

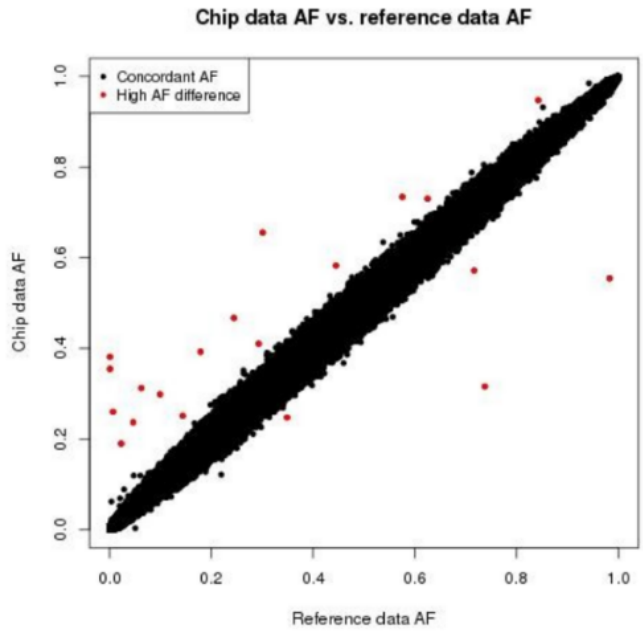
!! If needed, fix unsuccesful flipping as indicated in the following step (Step 5).

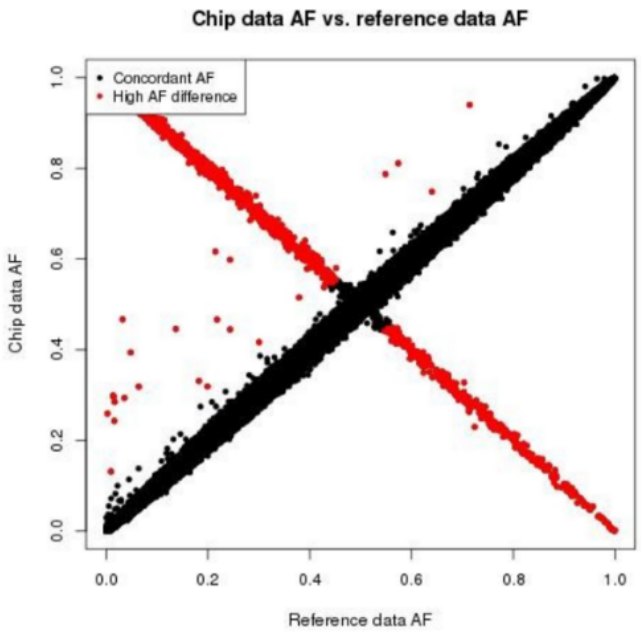
cmd COMMAND (compare_AF.R)

```
#!/bin/env Rscript --no-save

# Required packages
library(data.table) # For fast fread()

# Input variables
args <- commandArgs(TRUE)
indataset <- args[1]
ref_dataset <- args[2]

# Subset the dataset name
dataset_tag <- sub("_vcf_format.frq", "", indataset)

# Read in the frequency files
chip <- fread(indataset, header = T)
ref_data <- fread(ref_dataset, header = T)

# Take an intersection of the reference and chip data based on SNP column (in format CHR_PO
S_REF_ALT)
isec <- merge(ref_data, chip, by = "SNP")

# Check that AFs is within range of 10 pp in both datasets
af_ok <- abs(isec$AF.x - isec$AF.y) < 0.1

# Exclude those not within the AF range
exclude <- !af_ok

# Save the plot as jpg
jpeg(paste(dataset_tag, "_AF.jpg", sep=""))
# Plot first all and then excludable variants
plot(isec$AF.x, isec$AF.y, col=1, pch=20, main="Chip data AF vs. reference data AF", xlab="
Reference data AF", ylab="Chip data AF")
points(isec[exclude]$AF.x, isec[exclude]$AF.y, col=2, pch=20)
# Draw a legend
legend("topleft", legend=c("Concordant AF", "High AF difference"), col=c(1,2), pch=20, cex=
0.9)
dev.off()
```

EXPECTED RESULTS

Successful (left) and unsuccessful (right) genome build lift-over.





Optional additional allele swapping

Step 5.

If the previous step showed x-shaped plot, then some alleles are flipped incorrectly and those should be additionally corrected.

Use the **.strand** file and search for those alleles that were on - strand. Save the list of those alleles and flip them back.

Input files:

• <chip_strandfile>.strand

• <dataset>_b38.bed

• <dataset>_b38.bim

• <dataset>_b38.fam

Output files:

• flipped_in_update_build_script.txt

• <dataset>_b38_reflipped.bed

• <dataset>_b38_reflipped.bim

• <dataset>_b38_reflipped.fam

!! After running the commands below, confirm the results:

Re-run the previous steps **(Step 3** and **Step 4)** to compare the new chip data variant allele frequencies to the panel variant allele frequencies.

Now the data should not anymore contain variants with negative correlation.

cmd COMMAND

```
# Find the alleles that were on -
 strand and were thus incorrectly flipped with update_build.sh
cat <chip_strandfile>.strand | awk '{if ($5 == "-") print $0}' | cut -
f 1 > flipped_in_update_build_script.txt

# Flip the alleles back
plink --bfile <dataset>_b38 --flip flipped_in_update_build_script.txt --make-bed --
out <dataset>_b38_reflipped
```

Lift-over confirmation

Step 6.

Once you have successfully lifted over your chip data, verify the lift-over by checking variant positions and alleles.

For instance, compare the positions and alleles in your lifted-over data to known GRCh38/hg38 positions and alleles e.g. from dbSNP https://www.ncbi.nlm.nih.gov/SNP/.

Optional data consistency verification

Step 7.

If for some reason your **.fam** file does not include the sex information (e.g. a VCF converted to PLINK format files loses the sex information), sex information need to be added into the **.fam** file from a separate file.

Confirm that the sex information is present in the 5th column for all samples.

If not, obtain the sex information from the original source of your data and update the **.fam** file as suggested in the comman below.

Input files:

• <dataset>_genders.txt - containing FID and IID in the first two columns and sex information in the third column as 1 = male, 2 = female or 0 = ambiguous

• <dataset>_b38.bed

• <dataset>_b38.bim

• <dataset>_b38.fam

Output files:

• <dataset>_b38_updated_genders.bed

• <dataset>_b38_updated_genders.bim

• <dataset>_b38_updated_genders.fam

cmd COMMAND

```
# Update genders
plink --bfile <dataset>_b38 --update-sex <dataset>_genders.txt --make-bed --
out <dataset>_b38_updated_genders
```

Optional data consistency verification

Step 8.

Sometimes the alleles are marked for instance as '**B**', and such variants should be excluded.

Confirm that the alleles are in standard format i.e. **A**, **C**, **G** or **T**.

If not, correct the inconsistency by removing variants with non-standard allele notation as suggested in the command below.

Input files:

• <dataset>_b38_updated_genders.bed

• <dataset>_b38_updated_genders.bim

• <dataset>_b38_updated_genders.fam

Output files:

• <dataset>_b38_Bvariantlist.txt

• <dataset>_b38_clean.bed

• <dataset>_b38_clean.bim

• <dataset>_b38_clean.fam

cmd COMMAND

```
# Check your .bim file for B alleles
grep -P '\tB\t|\tB$' <dataset>_b38_updated_genders.bim | cut -
f 2 > <dataset>_b38_Bvariantlist.txt

# Remove those variants from the dataset
plink --bfile <dataset>_b38_updated_genders --exclude <dataset>_b38_Bvariantlist.txt --
make-bed --out <dataset>_b38_clean
```

Optional data consistency verification

Step 9.

Make sure no duplicate samples exist. Redundant individual IDs (IIDs) can cause some issues when working with VCF files (e.g. in the next step).

In case of duplicates, consider excluding them or adding a running number to those sample/individual IDs in the **.fam** file.

Input file:

• <dataset>_b38_clean.fam

Output file:

• <dataset>_b38_clean.fam

cmd COMMAND

```
# Find out possible sample ID (IID) duplicates
cut -d' ' -f 2,2 <dataset>_b38_clean.fam | sort | uniq -d

# If only few duplicates, edit <dataset>_b38_clean.fam
# and add a suffix to such IIDs.

# If many duplicates, append an index (row number) to each and every IID
cat <dataset>_b38_clean.fam | awk '{$2=$2"_"NR ; print $0}' >   <dataset>_b38_nodups.fam

# Rename the file back to match with the other PLINK files (.bed and .bim)
mv <dataset>_b38_nodups.fam <dataset>_b38_clean.fam
```

Optional VCF conversion

Step 10.

PLINK format files can be converted to VCF format if required.

Input files:

• <dataset>_b38_clean.bed

• <dataset>_b38_clean.bim

• <dataset>_b38_clean.fam

Output file:

• <dataset>_b38.vcf.gz

!! If the command fails to write the file, try removing 'bgz' from the command below, and then compress the output separately with for instance 'bgzip <dataset>_b38.vcf'

cmd COMMAND

```
# Convert the PLINK format to VCF format
plink --bfile <dataset>_b38_clean --recode vcf-iid bgz --output-chr M --out <dataset>_b38
```

ANNOTATIONS

Marita A. Isokallio 16 Apr 2018

In **Step10**, it is possible to use parameter '--output-chr chrM' instead of '--output-chr M' and omit **Step11**.

This will output chromosome names with 'chr' followed by number for autosomes or X/Y/XY/M for other chromosomes.
See PLINK documentation for more information:
https://www.cog-genomics.org/plink/2.0/data#irreg_output

Optional VCF conversion

Step 11.

Correct the chromosome notation to correspond the notation used in GRCh38/hg38.

GRCh38/hg38 chromosome notation uses 'chr' tag, whereas PLINK format uses only numbers.

First, generate a mapping file for the chromosome names as a space-separated file <old_name> <new_name>, one chromosome per line. Save the file as 'chr_names.txt'.

```
# Example of chr_names.txt format
1 chr1
2 chr2
...
X chrX
Y chrY
M chrM
```

Input files:

• chr_names.txt

• <dataset>_b38.vcf.gz

Output file:

• <dataset>_b38_renamed_chrs.vcf.gz

cmd COMMAND

```
# Correct the chromosome notation
bcftools annotate --rename-chrs chr_names.txt <dataset>_b38.vcf.gz -Oz -
o <dataset>_b38_renamed_chrs.vcf.gz
```