Genotype imputation workflow v3.0

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

Abstract

Citation: Kalle Pärn,Marita A. Isokallio,Javier Nunez Fontarnau,Aarno Palotie,Samuli Ripatti,Priit Palta Genotype imputation workflow v3.0. **protocols.io**
dx.doi.org/10.17504/protocols.io.nmndc5e
Published: 10 May 2018

Protocol

Requirements and preparatory steps

Step 1.

The actual imputation protocol begins at step 2.

All consecutive steps (commands given in 'cmd COMMAND' sections) must be run to ensure high-quality results.

For a 'quick and dirty' genotype imputation 'run', you can jump straight to **Steps 10-11** and only run these (assuming you already have all the required files in the correct format).

Throughout the protocol we assume Bash shell.

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

In the protocol, we assume human reference genome build version **GRCh38/hg38**. To lift-over and update the build of your chip data, you can use our lift-over protocol:
https://www.protocols.io/view/genotyping-chip-data-reference-genome-build-lift-o-nqtddwn

Please note that build 38 compatible VCF files should and auxiliary files might have **'**chr**'** prefixes in chromosome names. Therefore, some minor corresponding changes might be required in the related commands to parse/work with these files correctly.

If you prefer GRCh37/hg19 or other genome build versions, please download the corresponding reference genome, map and panel files and modify the suggested commands accordingly.

1.1 Software packages

1.1.1 Download and install the software packages

Required software packages are listed below with the versions used in this protocol. However, using the latest versions is recommended.

- BCFtools v1.7 http://www.htslib.org/download/
- R v3.4.1 https://www.r-project.org/
- R package data.table https://github.com/Rdatatable/data.table/wiki/Installation
- R package sm https://cran.r-project.org/package=sm
- Eagle v2.4 https://data.broadinstitute.org/alkesgroup/Eagle/
- Beagle v4.1 beagle.27Jan18.7e1.jar https://faculty.washington.edu/browning/beagle/beagle.html

1.1.2 Export the paths

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

```
echo
PATH=/path/to/installed/bcftools/executable/dir/:/path/to/installed/Rscript/e
xecutable/dir/:/path/to/installed/eagle/executable/dir/:/path/to/installed/be
agle/executable/dir/:$PATH >> $HOME/.bashrc
source $HOME/.bashrc
```

BCFtools plugin usage requires environment variable BCFTOOLS_PLUGINS exported, e.g:

```
echo export BCFTOOLS_PLUGINS=/path/to/bcftools/plugins >> $HOME/.bashrc
source $HOME/.bashrc
```

1.1.3 Install the R packages

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

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

```
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 genome and genetic map files

1.2.1 Fasta files

Homo Sapiens assembly hg38 version 0 is used and the required files are

- hg38_v0_Homo_sapiens_assembly38.fasta
- hg38_v0_Homo_sapiens_assembly38.fasta.fai

The files are available for downloading at Broad Insitute storage in Google cloud at https://console.cloud.google.com/storage/browser/broad-references/hg38/v0/?pli=1

If you prefer GRCh37/hg19, the reference genome files are available for downloading at Ensembl site at

https://grch37.ensembl.org/index.html

1.2.2. Genetic map files for phasing with Eagle

Genetic map file (all chromosomes in a single file) with recombination frequencies is downloaded together with Eagle for **GRCh38/hg38**.

- genetic_map_hg38_withX.txt.gz

We have processed the file according to the command below in order to split it per chromosome.

The resulting files are saved as:

- eagle_chr#_b38.txt (where # is the chromosome number)

```
# Take the lines for each chromosome separately,
# add a header row and store into a .txt file
for chr in {1..23}; do
    zcat genetic_map_hg38_withX.txt.gz | \
    grep ^$chr | sed '1ichr position COMBINED_rate(cM/Mb) Genetic_Map(cM)' \
    > eagle_chr${chr}_b38.map
done
```

Note: Currently the chromosome notation in the Eagle genetic map files is only the chromosome number without 'chr' and chrX is '23'.

Note: Starting from Eagle v2.4, also chromosome notation with 'chr' tag is supported.

If you prefer GRCh37/hg19, the genetic map file is available for downloading at Eagle download page at https://data.broadinstitute.org/alkesgroup/Eagle/downloads/tables/

- genetic_map_hg19_withX.txt.gz

1.2.3 Genetic map files for imputation with Beagle

We have processed the **GRCh38/hg38** Eagle genetic map files (provided by Eagle without 'chr' tag and chrX as '23') to be suitable for Beagle according to the commands below.

First, confirm the format of the downloaded Eagle genetic map files and make required changes such that the chromosome notation is with 'chr' tag and chromosome 23 is 'chrX'.

The resulting files are saved as:

- beagle_chr#_b38.map (where # is the chromosome number)

```
# For each Eagle genetic map file, re-order the columns, remove headers, add
'chr' and replace chr23 with chrX
for chr in {1..23}; do
        filename=eagle_chr${chr}_b38.map
        paste <(cut -d' ' -f1-2 $filename | sed '1d' | sed
's/^/chr/;s/chr23/chrX/') \
        <(cut -d' ' -f4 $filename | sed '1d') \
        <(cut -d' ' -f2 $filename | sed '1d') | \
        awk -v OFS='\t' '$2='.' {print}' > beagle_chr${chr}_b38.map
done
```

Note: Chromosome notation in the Beagle genetic map files is 'chr#' and chromosome 23 is 'chrX'.

If you prefer GRCh37/hg19, the genetic map file is available for downloading at Beagle site:

http://bochet.gcc.biostat.washington.edu/beagle/genetic_maps/

- plink.GRCh37.map.zip

1.3. Imputation reference panel files

1.3.1 Obtain the reference panel files

For increased imputation accuracy we recommend using a population-specific imputation reference panel, if available.

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

GRCh38/hg38 files are available at EBI 1000 genomes ftp site at ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/supporting/GRCh38_positions/

GRCh37/hg19 files are available at Beagle site already processed to be compatible with Beagle:

http://bochet.gcc.biostat.washington.edu/beagle/1000_Genomes_phase3_v5a/

The 1000 Genomes Project files can be downloaded for instance with command:

```
wget -np
ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/supporting/GRCh38_p
ositions/*
```

Save the phased reference panel VCF files per chromosome as follows (for consistency with the commands in the protocol):

- panel_phased_chr#.vcf.gz (where # is chromosome number)

NOTE: The reference panel files should contain non-missing, phased genotypes, chrX as diploid genotypes and all variants as biallelic records!

If your reference panel files are not in correct format, see some suggested commands below.

Also Eagle provides guidelines for processing the GRCh38/hg38 1000 Genomes Project files at: https://data.broadinstitute.org/alkesgroup/Eagle/#x1-320005.3.2

1.3.2 Check for multiallelic sites

Our v3.0 imputation workflow can also impute multiallelic sites, i.e. multiple alternative alleles at the same site, but these multiallelic variants MUST be **decomposed** (be split and represented as a set of biallelic variants) in the reference panel files.

Confirm that in your reference panel files, multiallelic sites (if present) 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 panel_phased_chr${chr}.vcf.gz -Oz -o
panel_phased_split_multiallelic_chr${chr}.vcf.gz
done
```

1.3.3 Check the chromosome notation in GRCh38/hg38 reference panel files

Confirm that the chromosome notation in your reference panel files follows the GRCh38/hg38 notations as 'chr#' for autosomal, 'chrX' for chromosome 23 and 'chrM' for mitochondrial sites. If not, adapt the example command below according to your files and rename the chromosomes:

```
# Create a list of all chromosome names in space-delimited format <old_name>
<new_name>
# Example of chr_names.txt file:
1 chr1
2 chr2
...
23 chrX
X chrX
MT chrM

bcftools annotate --rename-chrs chr_names.txt input.vcf.gz -Oz -o
output.vcf.gz
```

1.3.4 Generate a list of the reference panel sample IDs

List of sample IDs present in the reference panel, one line per sample ID, save with filename

- panel_sample_IDs.txt

The list of sample IDs can be generated from any of the VCF files (here chr22) as in the example below (assuming that all chromosomes contain the same set of samples):

```
bcftools query -l panel_phased_chr22.vcf.gz > panel_sample_IDs.txt
```

1.3.5 Reference panel allele frequencies

Generate a tab-delimited file of the reference panel allele frequencies, one variant per line, with columns CHR, SNP (in generated file header replaced with CHR_POS_REF_ALT), REF, ALT, AF (including the header line); note last for loop in following Bash script.

Use the phased reference panel VCF files as input with the example command below and save the file as

- panel.frq

```
# Check your reference panel VCF and if it does NOT contain AF in the INFO
field, calculate it with +fill-tags plugin
# Note: it requires environmental variable BCFTOOLS_PLUGINS exported (Step
1.2)
for chr in {1..23}; do
    bcftools +fill-tags panel_phased_chr${chr}.vcf.gz -Oz -o
panel_phased_chr${chr}_AF.vcf.gz -- -t AF
done

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

# Query the required fields from the VCF file 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'
panel_phased_chr${chr}.vcf.gz >> panel.frq
done
```

Note: Chromosome notation in the panel.frq file should follow the GRCh38/hg38 notations

('chr#' for autosomal chromosomes and 'chrX' for chromosome 23).

1.3.6 Create binary reference panel files

The phased reference panel files per chromosome are required in bref format (bref = binary reference. For more information see Beagle documentation at Beagle site:

https://faculty.washington.edu/browning/beagle/bref.16Dec15.pdf.

The required **bref.*.jar** is downloaded together with Beagle.

Use the phased reference panel VCFs as inputs for the example command below which saves the output files as:

- panel_phased_chr#.bref (where # is chromosome number)

```
# Convert each VCF to bref format
for chr in {1..23}; do
    java -jar bref.08Jun17.d8b.jar panel_phased_chr${chr}.vcf.gz
done
```

1.4. 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!

1.4.1 Input file:

Post-QC chip genotype data in VCFv4.2 format:

- <dataset>.vcf.gz

Note: Chromosome notation should follow the GRCh38/hg38 notations (e.g. 'chr#' for autosomal chromosomes, 'chrX', 'chrY' and 'chrM').

Note: If the input data was lifted over from an older genome build to version 38, cautious inspection of the data is highly recommended before proceeding with the protocol.

1.4.2 Final output files:

- <dataset>_imputed_info_chr#.vcf.gz (where # is chromosome number)
- <dataset>_postimputation_summary_plots.pdf

Note: Several intermediate files are created during the protocol. Those files can be used for troubleshooting and deleted once the successful imputation is confirmed.

ANNOTATIONS

Marita A. Isokallio 11 May 2018

Beagle authors have lately added genetic map files for GRCh38/hg38 as well.
The file `plink.GRCh38.map.zip` is available for downloading at Beagle site:
http://bochet.gcc.biostat.washington.edu/beagle/genetic_maps/

Marita A. Isokallio 31 May 2018

CORRECTION

Step 1.2.3:
Within the for loop awk command, single quotes should be replaced with double quotes as follows:
awk -v OFS='\t' '$2="." {print}' > beagle_chr${chr}_b38.map

Alternative to Step 1.2.3:
Download the genetic map files (plink.GRCh38.map.zip) provided at Beagle site:
http://bochet.gcc.biostat.washington.edu/beagle/genetic_maps/

Chip data validation and VCF formatting

Step 2.

Only autosomal and X chromosomes are kept for the imputation.

Input file:

• <dataset>.vcf.gz

Outpt file:

• <dataset>_chrfiltered.vcf.gz

Note: Confirm the results for example by comparing the input and output file sizes and line counts.

cmd COMMAND

```
# Generate a comma-separated string of chromosome names to keep
chrs=$(paste -d ' ' <(echo chr{1..22}) <(echo chrX) | tr ' ' ',')

# Keep only those wanted chromosomes
bcftools view -t $chrs <dataset>.vcf.gz -Oz -o <dataset>_chrfiltered.vcf.gz
```

paste parameter: -d delimiter bcftools view parameters: -t targets ^ exclusion prefix -Oz compressed output

Chip data validation and VCF formatting

Step 3.

Align the variant alleles to human reference genome to correct for any dataset-specific REF/ALT flips.

Ensure that only biallelic sites are kept in the chip data, as 'bcftools norm' may introduce false multiallelic sites.

Finally, replace the ID column with a 'SNP ID' in format CHROM_POS_REF_ALT ie. chromosome_position_<reference allele>_<alternative allele>.

Input files:

• <dataset>_chrfiltered.vcf.gz

• hg38_v0_Homo_sapiens_assembly38.fasta

Outpt file:

• <dataset>_SNPID.vcf.gz

Note: Confirm the results for example by comparing the input and output file sizes and line counts.

cmd COMMAND

```
# Align the alleles to the reference genome
bcftools norm -f hg38_v0_Homo_sapiens_assembly38.fasta -c ws <dataset>_chrfiltered.vcf.gz -
Ou | \
# Keep only biallelic records
bcftools view -m 2 -M 2 -Oz -o <dataset>_refcorrected.vcf.gz

# Replace the ID column with a CHR_POS_REF_ALT 'SNP ID'
bcftools annotate --set-id '%CHROM\_%POS\_%REF\_%ALT' <dataset>_refcorrected.vcf.gz -Oz -
o <dataset>_SNPID.vcf.gz
```

bcftools norm parameters: -f reference genome -c what to do when incorrect/missing REF allele is encountered: ws warn (w) and set/fix (s) bad sites bcftools view -m minimum number of alleles listed in REF and ALT columns -M maximum number of alleles listed in REF and ALT columns -Ou uncompressed output -Oz compressed output --threads number of threads to use for output compression bcftools annotate parameter: --set-id set ID column, % indicates a VCF field

Chip data validation and VCF formatting

Step 4.

Ensure that duplicate individuals do not exist in the chip data or between the chip and reference panel as this would compromise the imputation.

Input files:

• <dataset>_SNPID.vcf.gz

• panel_sample_IDs.txt

Outpt file:

• <dataset>_noduplicate_samples.vcf.gz

Note: Confirm the results for example by comparing the input and output file sizes and line counts.

cmd COMMAND

```
# Copy the panel sample ID file with a different name to the working directory
cp /path/to/panel_sample_IDs.txt duplicate_sample_IDs.txt

# Generate a list of sample IDs from the chip data, keep only duplicates and append to the
list of reference panel sample IDs
bcftools query -l <dataset>_SNPID.vcf.gz | uniq -d >> duplicate_sample_IDs.txt

# Remove the listed sample IDs from the chip data VCF
bcftools view -S ^duplicate_sample_IDs.txt --force-samples <dataset>_SNPID.vcf.gz -Oz -
o <dataset>_noduplicate_samples.vcf.gz
```

bcftools query parameters: -l list of sample IDs uniq -d only print duplicate lines bcftools view parameters: -S file of sample IDs to include ^ exclusion prefix --force-samples only warn about unknown subset of samples -Oz compressed output

Chip data validation and VCF formatting

Step 5.

Ensure that there are no duplicate variants (these might appear in some chip genotype datasets).

If duplicate variants are present, they need to be removed before imputation.

Input file:

• <dataset>_noduplicate_samples.vcf.gz

Outpt files:

• <dataset>_duplicate_variants.txt

• <dataset>_noduplicate_variants.vcf.gz

Note: Confirm the results for example by comparing the input and output file sizes and line counts.

cmd COMMAND

```
# Store a list of duplicate positions
bcftools query -f '%ID\n' <dataset>_noduplicate_samples.vcf.gz | uniq -
d > <dataset>_duplicate_variants.txt

# Check whether the file contains any variants
if [ -s <dataset>_duplicate_variants.txt ]; then
   # Then remove the duplicate variants
   bcftools view -
e ID=@<dataset>_duplicate_variants.txt <dataset>_noduplicate_samples.vcf.gz -Oz -
o <dataset>_noduplicate_variants.vcf.gz
else
  # If the file is empty i.e. no duplicate variants are present, only rename the file to be
 compatible with the next step
  mv <dataset>_noduplicate_samples.vcf.gz <dataset>_noduplicate_variants.vcf.gz
fi
```

bcftools query parameters: -f query fields % identicate the field uniq parameter: -d only print duplicate lines bcftools view parameters: -e exclusion with expression expression: ID=@file IDs included in the file -Oz compressed output

Chip data validation and VCF formatting

Step 6.

Exclude rare variants (redundant step if they are already removed in quality control steps taken before this protocol), and re-calculate the allele frequency to correctly represent the current samples in the dataset.

Input file:

• <dataset>_noduplicate_variants.vcf.gz

Outpt file:

• <dataset>_AF.vcf.gz

Note: Confirm the results for example by comparing the input and output file sizes and line counts.

cmd COMMAND

```
# Remove low allele count variants if they are not already removed
bcftools view -e 'INFO/AC<3 | (INFO/AN-INFO/AC)<3' <dataset>_noduplicate_variants.vcf.gz -
Ou |\
# Re-calculate allele frequency
bcftools +fill-tags  -Oz -o <dataset>_AF.vcf.gz -- -t AF
```

bcftools view parameters: -e exclude based on expression -Ou uncompressed output bcftools plugin syntax and parameters: +fill-tags re-calculates/adds INFO field tags -Oz compressed output -- separator for plugin-specific parameters -t define the tags to be re-calculated/added Alternatively, if all INFO field tags are wanted (see BCFtools documentation for complete list), remove the tag parameter: bcftools +fill-tags _noduplicate_variants.vcf.gz -Oz -o _AF.vcf.gz

Chip data validation and VCF formatting

Step 7.

Generate a frequency file for chip data and reference panel allele frequency comparison.

Input file:

• <dataset>_AF.vcf.gz

Outpt file:

• <dataset>_chip.frq

cmd COMMAND

```
# First generate a tab-delimited header for the allele frequency file
echo -e 'CHR\tSNP\tREF\tALT\tAF' > <dataset>_chip.frq

# Query the required fields from the VCF file and append to the allele frequency file
bcftools query -
f '%CHROM\t%ID\t%REF\t%ALT\t%INFO/AF\n' <dataset>_AF.vcf.gz >> <dataset>_chip.frq
```

echo parameter: -e enable interpretation of backslash escapes bcftools query parameter and syntax: -f format, where %field refers to a column in VCF file expression '%CHROM\t%ID\t%REF\t%ALT\t%INFO/AF\n' generates for each variant line tab-delimited format of: CHR, CHR_POS_REF_ALT, REF, ALT, AF

Chip data validation and VCF formatting

Step 8.

Compare the chip data and reference panel allele frequencies and generate an exclusion list of those variants where AF values differ more than 10 pp.

Download the R script given in 'FILE' link at the end of this step and save it as 'plot_AF.R'.

Set the script as executable for instance with 'chmod +x plot_AF.R' before running it for the first time.

Run the script as indicated in the example command by giving the two frequency files as input arguments.

Input files:

• <dataset>_chip.frq

• panel.frq

Outpt files:

• <dataset>_AFs.jpg

• <dataset>_AF_hist.jpg

• <dataset>_exclude.txt

!! Inspect the generated plots before proceeding to the next step:

See in the expected results section at the end of this step, for examples of plots from a high-quality dataset.

• <dataset>_AFs.jpg

Comparison of the chip AF values to the panel AF values. Ideally, it should show a quite tight, uniform diagonal line of increasing slope (from bottom left to top right).

Variants with AF values differing more than 10 pp are marked with red color and included to the exclusion list.

• <dataset>_AF_hist.jpg

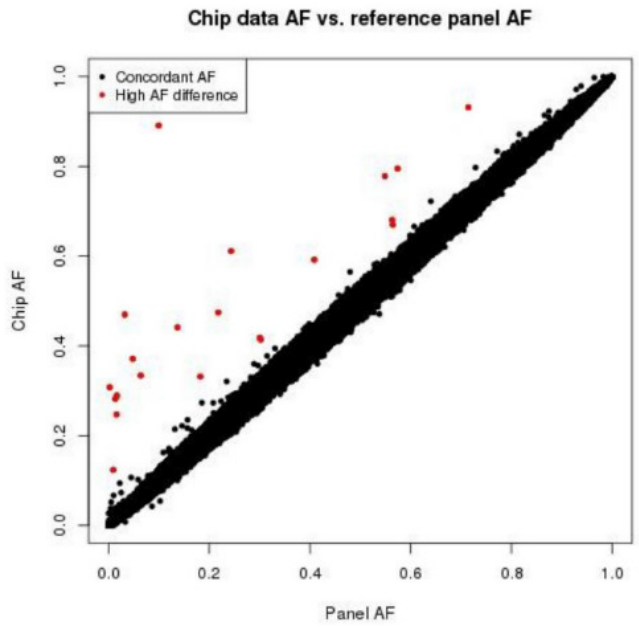
Distribution of the AF values. Ideally the histogram should be smooth and skewed toward the smallest AFs.

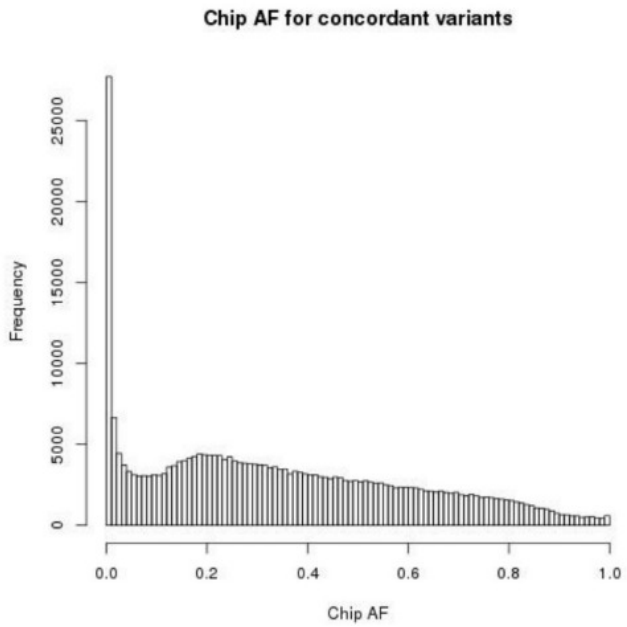
cmd COMMAND (plot_AF.R)

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

# Genotype imputation protocol v3 - pre-imputation allele frequency plots
# Written by Kalle Pärn, Marita A. Isokallio, Paavo Häppölä, Javier Nunez Fontarnau and Pri
it Palta

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

# Input variables
args <- commandArgs(TRUE)
indata <- args[1]
paneldata <- args[2]

# Read in the frequency files
chip <- fread(indata, header = T)
panel <- fread(paneldata, header = T)

# Generate a dataset tag
indataset <- sub("_chip.frq", "", indata)

# Take an intersection of the panel and chip data based on SNP column (in format CHR_POS_RE
F_ALT)
isec <- merge(panel, 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(indataset, "_AFs.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 panel AF", xlab=
"Panel AF", ylab="Chip 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()

# Save the plot as jpg
jpeg(paste(indataset, "_AF_hist.jpg", sep = ""))
# Chip AF histogram for concordant AF variants
hist(isec[!exclude]$AF.y, breaks=100, main="Chip AF for concordant variants", xlab="Chip AF
")
dev.off()

# Write out the exclusion list
write.table(isec[exclude]$SNP, paste(indataset, "_exclude.txt", sep=""), quote=F, row.names
=F, col.names=F)
```

EXPECTED RESULTS

Examples of <dataset>_AFs.jpg and <dataset>_AF_hist.jpg plots.





Chip data validation and VCF formatting

Step 9.

Exclude the variants with highly discordant allele frequencies as listed in the previous step.

Input files:

- <dataset>_exclude.txt (output from step8)
- <dataset>_AF.vcf.gz (output from step 6)

Outpt files:

- <dataset>_exclude_sorted.txt
- <dataset>_for_phasing.vcf.gz

Note: Confirm the results for example by comparing the input and output file sizes and line counts.

cmd COMMAND

```
# Sort the exclusion list
sort -V <dataset>_exclude.txt > <dataset>_exclude_sorted.txt

# Exclude the variants
bcftools view -e ID=@<dataset>_exclude_sorted.txt <dataset>_AF.vcf.gz -Oz -
o <dataset>_for_phasing.vcf.gz
```

bcftools view parameters: -e exclusion expression: ID=@file IDs in indicated file -Oz compressed output

Chip data pre-phasing

Step 10.

Chip data pre-phasing will speed up the genotype imputation step.

Phase haplotypes for each chromosome separately before imputation.

Input files:

• <dataset>_for_phasing.vcf.gz

• eagle_chr#_b38.map (where # is chromosome number)

Outpt file:

• <dataset>_for_imputation_chr#.vcf.gz (where # is chromosome number)

Note: Confirm the results for example by comparing the input and output file sizes and line counts.

cmd COMMAND

```
# Run phasing for each chromosome
for chr in {1..23}; do

    # Chromosome 23 is coded as 'chrX' in Eagle, whereas autosomal chromosomes are chromoso
me numbers alone
    if [ "$chr" == "23" ]; then
          chrname=chrX
    else
          chrname=$chr
    fi

    # Run phasing for each chromosome separately
    eagle \
           --vcf <dataset>_for_phasing.vcf.gz \
           --chrom ${chrname} \
           --geneticMapFile eagle_chr${chr}_b38.map \
           --numThreads=8 \
           --Kpbwt=20000 \
           --outPrefix <dataset>_for_imputation_chr${chr}
done
```

The command here is split on multiple lines for better readability. However, if the command is copied from here, spaces/tabs may not be correct and cause an error. In case of errors, first try to reformat the command to a single line. eagle parameters: --vcf VCF format input containing the genotypes --chrom chromosome to analyze --geneticMapFile HapMap genetic map provided with eagle download (here, uncompressed and separated per chromosome) --numThreads number of threads --Kpbwt number of conditioning haplotypes --outPrefix prefix for output files

Genotype imputation

Step 11.

Run genotype imputation for each chromosome separately.

Input files:

• <dataset>_for_imputation_chr#.vcf.gz

• panel_phased_chr#.bref (where # is chromosome number)

• beagle_chr#_b38.map (where # is chromosome number)

Outpt file:

• <dataset>_imputed_chr#.vcf.gz (where # is chromosome number)

cmd COMMAND

```
# Run imputation for each chromosome
for chr in {1..23}; do

    java -Xss5m -Xmx16g -jar beagle.jar \
        gt=<dataset>_for_imputation_chr${chr}.vcf.gz \
        ref=panel_phased_chr${chr}.bref \
        map=beagle_chr${chr}_b38.map \
        out=<dataset>_imputed_chr${chr} \
        nthreads=16 \
        niterations=10 \
        ne=20000 \
        impute=true \
        gprobs=true \
        seed=-99999
done
```

The command here is split on multiple lines for better readability. However, if the command is copied from here, spaces/tabs may not be correct and cause an error. In case of errors, first try to reformat the command to a single line. beagle parameters: gt - a VCF file containing the genotypes ref - a VCF file containing the phased reference genotypes map - PLINK format genetic map on the cM scale out - output file prefix nthreads - number of threads niterations - number of phasing iterations ne - effective population size when imputing ungenotyped markers impute - whether markers that are present in the reference panel but absent in your data will be imputed gprobs - whether a GP format field (genotype probability) will be included in the output VCF file when imputing ungenotyped markers seed - seed for the random number generator

Post-imputation processing and quality assurance

Step 12.

Post-imputation processing

Re-calculate/add INFO field values and compute Impute2-like INFO scores for each variant.

Input file:

• <dataset>_imputed_chr#.vcf.gz (where # is chromosome number)

Outpt files:

• <dataset>_imputed_chr#.vcf.gz.tbi (where # is chromosome number)

• <dataset>_imputed_info_chr#.vcf.gz (where # is chromosome number)

cmd COMMAND

```
# Re-calculate and add INFO field values for each chromosome
for chr in {1..23}; do

    # Create index file
    bcftools index -t <dataset>_imputed_chr${chr}.vcf.gz

        # Re-calculate allele frequency and compute Impute2-like INFO score
    bcftools +fill-tags <dataset>_imputed_chr${chr}.vcf.gz -Ou -- -t AF | \
        bcftools +impute-info -Oz -o <dataset>_imputed_info_chr${chr}.vcf.gz

done
```

bcftools plugins syntax and parameters: +fill-tags re-calculates/adds INFO field tags (see BCFtools documentation for complete list) -- separator for plugin-specific parameters -t define the tags to be re-calculated/added +impute-info computes Impute2-like INFO score -Ou uncompressed output -Oz compressed output Alternatively, if all INFO field tags are wanted, remove the plugin-specific parameter: bcftools +fill-tags _imputed_chr${chr}.vcf.gz -Ou

Post-imputation processing and quality assurance

Step 13.

Imputation QA

Extract allele frequencies (AF) and INFO scores from the imputed VCF file.

Group the variants by their AF into either of the following categories,

1: AF >= 5% or AF <= 95% (common variants)

2: 0.5% <= AF < 5% or 95% < AF <= 99.5% (low-frequency variants)

3: AF < 0.5% or AF > 99.5% (rare variants)

Input file:

• <dataset>_imputed_info_chr#.vcf.gz (where # is chromosome number)

Outpt file:

• <dataset>_chr#_varID_AF_INFO_GROUP.txt (where # is chromosome number)

cmd COMMAND

```
# Generate an allele frequency file for plotting for each chromosome
for chr in {1..23}; do

    # Generate a header for the output file
    echo -
e 'CHR\tSNP\tREF\tALT\tAF\tINFO\tAF_GROUP' > <dataset>_chr${chr}_varID_AF_INFO_GROUP.txt

    # Query only the required fields and add allele frequency group (1, 2 or 3) as the ast
column
    bcftools query -
f '%CHROM\t%CHROM\_%POS\_%REF\_%ALT\t%REF\t%ALT\t%INFO/AF\t%INFO/INFO\t-
\n' <dataset>_imputed_info_chr${chr}.vcf.gz |\
    # $5 refers to AF values, $7 refers to AF group
    awk -v OFS="\t" \
        '{if ($5>=0.05 && $5<=0.95) $7=1; \
          else if(($5>=0.005 && $5<0.05) || ($5<=0.995 && $5>0.95)) $7=2; \
          else $7=3} \
        { print $1, $2, $3, $4, $5, $6, $7 }' \
    >> <dataset>_chr${chr}_varID_AF_INFO_GROUP.txt

done
```

The command here is split on multiple lines for better readability. However, if the command is copied from here, spaces/tabs may not be correct and cause an error. In case of errors, first try to reformat the command to a single line. echo parameter: -e enable interpretation of backslash escapes bcftools query parameters: -f format, where %field refers to a column in VCF file expression '%CHROM\t%CHROM_%POS_%REF_%ALT_%INFO/AF\t%INFO/INFO\t-\n' generates for each variant line tab-delimited format of: CHR, CHR_POS_REF_ALT, AF, INFO, - awk parameter and syntax: -v OFS output field separator see if the AF ($5) is within a range and set the group number accordingly to the last column ($7) print all the columns and append to the output file

Post-imputation processing and quality assurance

Step 14.

Imputation QA

Produce multiple plots to illustrate the distribution of Impute2-like INFO scores and allele frequencies of the imputed data, as well as the comparison of allele frequencies between the imputed data and the reference panel.

Download the R script given in 'FILE' link at the end of this step and save it as 'plot_INFO_and_AF_for_imputed_chrs.R'.

Set the script as executable for instance with 'chmod +x plot_INFO_and_AF_for_imputed_chrs.R' before running it for the first time.

Run the script as indicated in the example command by giving the dataset name and panel frequency files as input arguments.

Input files:

• panel.frq

• <dataset>_chr#_varID_AF_INFO_GROUP.txt (where # is chromosome number)

Outpt files:

• <dataset>_chr#_postimputation_summary_plots.png (where # is chromosome number)

• <dataset>_postimputation_summary_plots.pdf

!! NOTE: If the number of variants in your dataset is very small, modify the 'plot_INFO_and_AF_for_imputed_chrs.R' script on line 30-31 to contain all variants.

!! Inspect the plots:

See example of the produced plots in expected results section a the end of this step.

Top left:

Impute2-like INFO score density for each AF group.

Bottom left:

AF distribution of the imputed chip data.

Top right:

AF comparison between the panel and imputed chip data.

Bottom right:

Absolute AF difference along the chromosome positions between the panel and imputed chip data.

cmd COMMAND (plot_INFO_and_AF_for_imputed_chrs.R)

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

# Genotype imputation protocol v3 - post-imputation QA plots
# Written by Kalle Pärn, Marita A. Isokallio, Paavo Häppölä, Javier Nunez Fontarnau and Pri
it Palta

# Required packages
library(data.table) # for fast fread()
library(sm)  # for density plotting

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

# Reference panel frequency file
panel <- fread(paneldata, header = T)

# Generate plots and save as a png file per chromosome
for (chr in 1:23) {
        print(paste("Working on chr ", chr, " now..."))

        # Creates the filename as <dataset>_chr#_postimputation_summary_plots.png
        png(filename = paste(indataset, "_chr", chr, "_postimputation_summary_plots.png",
sep = ""), width = 1200, height = 1200, unit = "px")

        # Set the plots as two rows and columns
        par(mfrow = c(2,2))
        par(cex.axis = 1.6, cex.lab = 1.5, cex.main = 1.6)

        # Read in data for a file <dataset>_chr#_varID_AF_INFO_GROUP.txt
        imp_vars <-
 fread(paste(indataset, "_chr", chr, "_varID_AF_INFO_GROUP.txt", sep = ""), header = TRUE)

        # Create random sample for INFO score plot instead of plotting all values
        rand1 <- sample(1:dim(imp_vars)[1], 100000, replace = FALSE)
        temp1 <- imp_vars[rand1,]

        # Merge by common variants, SNP column in format CHR_POS_REF_ALT
        isec <- merge(panel, imp_vars, by="SNP")

        # Plot INFO score distributions
        sm.density.compare(temp1$INFO, temp1$AF_GROUP, xlab = "Impute2-
like INFO score", ylab = "Density", col = col, lty = rep(1,3), lwd = 3, xlim = c(0,1), cex
= 1.4, h = 0.05)
        title(paste("Imputation of chr", chr, " variants", sep = ""))
        legend("topleft", legend = c("MAF > 5%", "MAF 0.5-5%", "MAF < 0.5%"), lty = c(1, 1
, 1), lwd = c(2.5, 2.5, 2.5), col = c("red", "green3", "blue"))

        # Plot AF of panel vs. imputed variants
        plot(isec$AF.x, isec$AF.y, col = 1, pch = 20, main = "Imputed AF vs. reference pan
el AF", xlab = "Reference panel AF", ylab = "Imputed AF", xlim = c(0,1), ylim = c(0,1))

        # Imputed data AF histogram for intersecting variants
        hist(isec$AF.y, breaks = 200, main = "AF distribution of imputed variants", xlab =
 "Imputed AF")
```

```
        # Plot absolute AF difference as imputed AF - panel AF
        isec$POS <-
 as.numeric(as.character(data.frame(do.call('rbind', strsplit(as.character(isec$SNP),'_',fi
xed=TRUE)))[,2]))
        # Order the variants by position
        sisec <- isec[order(isec$POS),]
        plot(sisec$POS, sisec$AF.y -
 sisec$AF.x, main = "Absolute AF differences along the chromosome", col = rgb(0,100,0,50, m
axColorValue=255), ylim = c(-0.1, 0.1), pch = 16, xlab = "Chromosome position", ylab = "AF
difference (imputed - panel)")

        # Close the png
        dev.off()
}
```

EXPECTED RESULTS

Example of chromosome 2 post-imputation summary plots.