Apr 01, 2019 Version 2

Genotype imputation workflow v3.0 V.2

DOI
dx.doi.org/10.17504/protocols.io.xbgfijw

Kalle Pärn[1], Marita A. Isokallio[1], Javier Nunez Fontarnau[1], Aarno Palotie[2,1], Samuli Ripatti[2,1], Priit Palta[2,1]

[1]FIMM, University of Helsinki; [2]equal contribution

FIMM HumGen Sequenc...

Priit Palta
FIMM, University of Helsinki

DOI: dx.doi.org/10.17504/protocols.io.xbgfijw

Protocol Citation: Kalle Pärn, Marita A. Isokallio, Javier Nunez Fontarnau, Aarno Palotie, Samuli Ripatti, Priit Palta 2019. Genotype imputation workflow v3.0. **protocols.io** **https://dx.doi.org/10.17504/protocols.io.xbgfijw**

Protocol status: Working

Created: January 21, 2019

Last Modified: April 01, 2019

Protocol Integer ID: 19528

Keywords: genotype imputation, imputation, Beagle, haplotype phasing, pre-phasing, Eagle

Requirements and preparatory steps

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 9-10** 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 reference panel files and modify the suggested commands accordingly.

1.0 Docker images and preprepared files
We have prepared docker images to ease up the preparatory steps depending on your application:

- genotype-liftover-imputation-protocols-light
- genotype-liftover-imputation-protocols-full

Light image contains the software packages installed (Step 1.1), the reference genome (Step 1.2.1), genetic map files (Steps 1.2.2 and 1.2.3) and example scripts to run commands in each step (Steps 2-13).
Full image contains the software packages installed (Step 1.1), preprepared publically available reference files (as defined in Steps 1.2 and 1.3) and example scripts to run commands in each step (Steps 2-13).

Additional to this protocol, the images contain software packages and files required for genotyping chip data lift-over protocol version 2: https://www.protocols.io/view/genotyping-chip-data-lift-over-to-reference-genome-xbhfij6

The docker images are available at Docker hub
To pull the image, use the command below and replace <tag> with 'light' or 'full' according to which image you want:

Command

```
docker pull seqinfoteam/genotype-liftover-imputation-protocols:<tag>
```

!! If you don't want to use Docker, you can install the software packages by yourself (Step 1.1), and either download preprepared files (below) or prepare the files by yourself (Step 1.2 and 1.3).

Preprepared reference data (included into the docker images as listed above) are also available for separate downloading at:
https://console.cloud.google.com/storage/browser/fimm-public-data/Imputation/

And example scripts how to run each step at:
https://console.cloud.google.com/storage/browser/fimm-public-data/Imputation/dockers/genotype-imputation-protocol/

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 (or later version) http://www.htslib.org/download/
- R v3.4.1 (or later version) 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 v2.3.5 https://data.broadinstitute.org/alkesgroup/Eagle/
- Beagle v4.1 beagle.27Jan18.7e1.jar https://faculty.washington.edu/browning/beagle/b4_1.html
- Beagle bref bref.27Jan18.7e1.jar https://faculty.washington.edu/browning/beagle/bref.27Jan18.7e1.jar

1.1.2 Export the paths

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

Command

```
echo PATH=/path/to/bcftools/:/path/to/R/:/path/to/eagle/:$PATH \
>> $HOME/.bashrc

source $HOME/.bashrc
```

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

Command

```
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.:

Command

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

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:

- Homo_sapiens_assembly38.fasta
- 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 for **GRCh38/hg38** are available for downloading at Eagle download page at:
https://data.broadinstitute.org/alkesgroup/Eagle/downloads/tables/

- genetic_map_hg38_withX.txt.gz

We have processed the file according to the command below in order to split it per chromosome with correct headers.
The resulting files are saved as:

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

Command

```
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'. 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

Genetic map files fro Beagle are available for downloading at Beagle download page at http://bochet.gcc.biostat.washington.edu/beagle/genetic_maps/

- plink.GRCh38.map.zip

Unzip the files and change the chromosome notation from PLINK format (only number or X) to GRCh38/hg38 standard notation with 'chr' tag as follows:

Command

```
# Unzip the files
unzip plink.GRCh38.map.zip

# Rename chromosome 23
mv plink.chrX.GRCh38.map plink.chr23.GRCh38.map

# Add 'chr' tag to the beginning of the line and
# store the output with suitable filename
for CHR in {1..23}; do
    cat plink.chr${CHR}.GRCh38.map | \
    sed 's/^/chr/' \
    > beagle_chr${CHR}_b38.map
done
```

Note: For GRCh38/hg38, the 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 (1000 GP) (**www.nature.com/articles/nature15393**) data can be used instead.

GRCh38/hg38 files are available at EBI 1000 genomes ftp site:
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/b37.bref/

The 1000 GP files can be downloaded for instance with command:

Command

```
wget
http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/supporting/
GRCh38_positions/ALL.chr{{1..22},X}_GRCh38.genotypes.20170504.vcf.gz{,
.tbi}
```

NOTE:
The reference panel files should contain:

- **phased genotypes,**
- **chromosome names with 'chr' and chromosome X as 'chrX',**
- **all variants as biallelic records,**
- **only SNPs and INDELs,**
- **only unique variants,**
- **non-missing data,**
- **chrX as diploid genotypes, and**
- **only unique IDs**

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

1.3.2 Minimum quality control

Here, we have piped most of the processing steps together in order to save significant amount of time by avoiding writing out multiple intermediate files. If your imputation reference panel does not require all the steps, modify the command accordingly.

For 1000GP data:

1. Generate a text file containing space-spearated old and new chromosome names. This is required to rename the numerical chromosome names with 'chr' tag. Apply the new chromosome names with 'bcftools annotate'.

2. Remove the rare variants, here singletons and doubletons by setting AC threshold with 'bcftools view'.
3. Split multiallelic sites to biallelic records with 'bcftools norm'.
4. Keep only SNPs and INDELs with 'bcftools view'. Here, the 1000GP data included a tag VT in the INFO field and data contain also structural variants which should be excluded.
5. Align the variants to reference genome with 'bcftools norm' in order to have the REF and ALT alleles in the shortest possible representation and to confirm that the REF allele matches the reference genome, additionally remove duplicate variants (-d none).
6. After alignment, remove multiallelic records with 'bcftools view', since these are formed during the alignment if the REF does not match with the reference genome.
7. Finally, remove sites containing missing data with 'bcftools view'.

Command

```
# Generate a chromosome renaming file
for CHR in {1..23} X ; do
    echo ${CHR} chr${CHR}
done >> chr_names.txt

# Multiple processing commands piped together
for CHR in {1..22} X; do
    bcftools annotate --rename-chrs chr_names.txt \
        ALL.chr${CHR}_GRCh38.genotypes.20170504.vcf.gz -Ou | \
    bcftools view -e 'INFO/AC<3 | INFO/AN-INFO/AC<3' -Ou | \
    bcftools norm -m -any -Ou | \
    bcftools view -i 'INFO/VT="SNP" | INFO/VT="INDEL"' -Ou | \
    bcftools norm -f Homo_sapiens_assembly38.fasta -d none -Ou | \
    bcftools view -m 2 -M 2 -Ou | \
    bcftools view -g ^miss -Oz -o 1000GP_chr${CHR}.vcf.gz"
done
```

If multiallelic sites are present in your data, in order to preserve them throughout the protocol, set ID field with unique IDs e.g. in format CHR_POS_REF_ALT. (RSIDs might contain duplicates, when the multiallelic sites are decomposed.)

Command

```
for CHR in {1..23}; do
    bcftools annotate \
    --set-id '%CHROM\_%POS\_%REF\_%ALT' \
    panel_chr${CHR}.vcf.gz  \
    -Oz -o panel_SNPID_chr${CHR}.vcf.gz
done
```

1.3.3 Convert haploid genotypes to homozygous diploids
Often chrX is represented as haploid genotypes for males, however, Beagle can only handle diploid genotypes.
The command here produces unphased diploid genotypes. But since the haploid genotypes are in diploid format as REF/REF or ALT/ALT, we can simply set the phase for those alleles with a simple sed replacement.

The chrX ploidy can be corrected as follows:

Command

```
# Fix the chromosome X ploidy to phased diploid
# Requires a ploidy.txt file containing
# space-separated CHROM,FROM,TO,SEX,PLOIDY
echo "chrX 1 156040895 M 2" > ploidy.txt
bcftools +fixploidy \
    1000GP_chrX.vcf.gz -Ov -- -p ploidy.txt | \
    sed 's#0/0#0\|0#g;s#1/1#1\|1#g' | \
bcftools view -Oz -o 1000GP_chr23.vcf.gz"
```

1.3.4 Duplicate ID removal
Remove duplicate IDs. If you wish to preserve all multiallelic sites, replace the ID column with a unique ID e.g. CHR_POS_REF_ALT (as indicated in **Step 1.3.2**).

Here, 1000GP did not contain multiallelic sites after AC filtering, and thus, RSIDs were preserved in the ID column. And since RSIDs are not always unique, duplicates should be removed.

Command

```
for CHR in {1..23}; do
    bcftools query -f '%ID\n' 1000GP_chr${CHR}.vcf.gz | \
    sort | uniq -d > 1000GP_chr${CHR}.dup_id

    if [[ -s 1000GP_chr${CHR}.dup_id ]]; then
        bcftools view -e ID=@1000GP_chr${CHR}.dup_id \
        1000GP_chr${CHR}.vcf.gz \
        -Oz -o 1000GP_filtered_chr${CHR}.vcf.gz
    else
        mv 1000GP_chr${CHR}.vcf.gz \
        1000GP_filtered_chr${CHR}.vcf.gz
    fi
done
```

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 format CHR_POS_REF_ALT), REF, ALT, AF (including the header line).

First, update (or add) AF values in the INFO field, calculate it with BCFtools plugin +fill-tags:

Command

```
# Check if the VCF does NOT contain AF in the INFO field,
# and calculate it with bcftools +fill-tags plugin

for CHR in {1..23}; do
    bcftools +fill-tags \
        1000GP_filtered_chr${CHR}.vcf.gz \
        -Oz -o 1000GP_AF_chr${CHR}.vcf.gz -- -t AF
done
```

Extract the wanted fields from each VCF file and combine as a single output file with the header:

Command

```
# Generate a tab-delimited header
echo -e 'CHR\tSNP\tREF\tALT\tAF' \
    > 1000GP_imputation_all.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' \
    1000GP_AF_chr${CHR}.vcf.gz \
    >> 1000GP_imputation_all.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 from Beagle site:

Command

```
wget
https://faculty.washington.edu/browning/beagle/bref.27Jan18.7e1.jar
```

Use the processed imputation reference panel VCFs as inputs for the example command below.
The output files have the suffix '.bref' instead of '.vcf.gz'.

Command

```
# Convert each file to bref format
for CHR in {1..23}; do
    java -jar /path/to/bref.27Jan18.7e1.jar \
        1000GP_AF_chr${CHR}.vcf.gz
done
```

1.3.7 Generate a list of the reference panel sample IDs
List of sample IDs present in the reference panel, one line per sample ID can be generated from any of the VCF files as in the example below (assuming that all chromosomes contain the same set of samples):

Command

```
bcftools query -l 1000GP_AF_chr22.vcf.gz \
    > 1000GP_sample_IDs.txt
```

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 and chrX genotypes as diploid genotypes:

- 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 build version 38, cautious inspection of the data is highly recommended before proceeding with the protocol.

Note: If chrX is represented as haploid genotypes, follow step 1.3.3 first 'bcftools +fixploidy' command (not the other two piped commands) to convert to diploid genotypes.

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.

Chip data validation and VCF formatting

2 Only autosomal and X chromosomes are kept for the imputation.

Input file:

- DATASET.vcf.gz (your QC'ed genotyping chip data in VCFv4.2 format, chrX as diploid genotypes)

Outpt file:

- DATASET_chrfiltered.vcf.gz

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

Command

bcftools view parameters:
-t targets
-Oz compressed output

```
DATASET=your_dataset_prefix
# A comma-separated string of chrs to keep
chrs=$(echo chr{1..22} chrX | tr ' ' ',')

# Keep only those wanted chromosomes
bcftools view -t $chrs ${DATASET}.vcf.gz \
    -Oz -o ${DATASET}_chrfiltered.vcf.gz
```

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 if the REF does not match to the reference genome.

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

Input files:

- DATASET_chrfiltered.vcf.gz (**Step 2**)

- Homo_sapiens_assembly38.fasta or another wanted reference genome file (**Step 1.2.1**)

Outpt file:
- DATASET_SNPID.vcf.gz

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

Command

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
bcftools annotate parameter:
--set-id set ID column, % indicates a VCF field

```
DATASET=your_dataset_prefix
FASTA=/path/to/your/reference_genome.fasta

# Align the alleles to the reference genome,
# and keep only biallelic records
bcftools norm -f ${FASTA} -c ws \
    ${DATASET}_chrfiltered.vcf.gz -Ou | \
bcftools view -m 2 -M 2 \
    -Oz -o ${DATASET}_refcorrected.vcf.gz

# Replace the ID column with a CHR_POS_REF_ALT
bcftools annotate \
    --set-id '%CHROM\_%POS\_%REF\_%ALT' \
    ${DATASET}_refcorrected.vcf.gz \
    -Oz -o ${DATASET}_SNPID.vcf.gz
```

4 Ensure that duplicate individuals do not exist in the chip data or between the chip and the imputation reference panel. Furthermore, ensure that there are no duplicate variants (these might appear in some chip genotype datasets and after reference genome alignment). Both sample and variant duplicates would compromise the imputation.

Input files:

- DATASET_SNPID.vcf.gz (**Step 3**)
- panel_sample_IDs.txt (**Step 1.3.7**)

Outpt file:

- DATASET_noduplicate_variants.vcf.gz

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

Command

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
-Ou uncompressed output
+fill-tags plugin specific parameters after --
-t tags to be updated, here AC, AN and AF
-Oz compressed output

```
DATASET=your_dataset_prefix
PANEL_ID=panel_sample_IDs.txt

# Copy the panel sample ID file with
# a different name to the working directory
cp ${PANEL_ID} duplicate_sample_IDs.txt

# Generate a list of chip data sample IDs,
# keep only duplicates and
# append to the list of panel sample IDs
bcftools query -l ${DATASET}_SNPID.vcf.gz | \
    uniq -d >> duplicate_sample_IDs.txt

# Exclude duplicate samples and MAC=0 variants,
# and update AF value after sample removals.
# Finally, exclude duplicate variants
bcftools view -S ^duplicate_sample_IDs.txt \
   --force-samples ${DATASET}.vcf.gz -Ou | \
bcftools +fill-tags -Ou -- -t AC,AN,AF | \
bcftools norm -d none \
    -Oz -o ${DATASET}_noduplicate_variants.vcf.gz
```

5 Exclude rare variants (redundant step if they are already removed in quality control steps taken before this protocol).

Input file:

- DATASET_noduplicate_variants.vcf.gz (**Step 4**)

Outpt file:

- DATASET_AF.vcf.gz

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

Command

bcftools view parameters:
-e exclude based on expression
-Oz compressed output

```
DATASET=your_dataset_prefix

# Remove low allele count variants
# (redundant if they are already removed)
bcftools view -e 'INFO/AC<3 | (INFO/AN-INFO/AC)<3' \
    ${DATASET}_noduplicate_variants.vcf.gz \
    -Oz -o ${DATASET}_AF.vcf.gz
```

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

Input file:

- DATASET_AF.vcf.gz (**Step 5**)

Outpt file:

- DATASET_chip.frq

Command

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 a tab-delimited format of:
CHR, CHR_POS_REF_ALT, REF, ALT, AF

```
DATASET=your_dataset_prefix

# Generate a tab-delimited header
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
```

7 Compare the chip data and the imputation reference panel allele frequencies and generate an exclusion list of those variants where AF values differ more than 10 pp or their log2 fold change is >5 or <-5.

Variants showing highly discrepant AF frequency will be excluded (in the next step). Confirm from the plot that number of concordant variants is high (preferably >300k) to ensure successful phasing (in **Step 9**).

Copy and save the R script below 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 below.

Here, we use 0.1 as a threshold for AF difference and 5 for the log2 fold change e.g. if chip data AF differs more than 10 pp from the reference data AF, it is marked as discordant. However, less conservative values may be more suitable if a population specific reference data is not available.

Input files:
- DATASET_chip.frq (**Step 6**)
- panel.frq (**Step 1.3.5**)

Outpt files:
- DATASET_AFs.png
- DATASET_exclude.txt
- DATASET_nonpanel_exclude.txt

Additional output files:
- DATASET_panel_isec.txt
- DATASET_discrepant_AF.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.

Left plot
Comparison of the chip AF values to the panel AF values. Ideally, it should show a quite tight, uniform diagonal line of increasing slope 1. Variants with AF values differing more than 10 pp or log2 fold change >5 or <-5 are marked with red color and included into the exclusion list. See also the number of non-panel variants which should not be high in comparison to concordant variant count. with a population-specific

Right plot
Distribution of the AF values. Ideally the histogram should be smooth and skewed towards the smallest AF values.

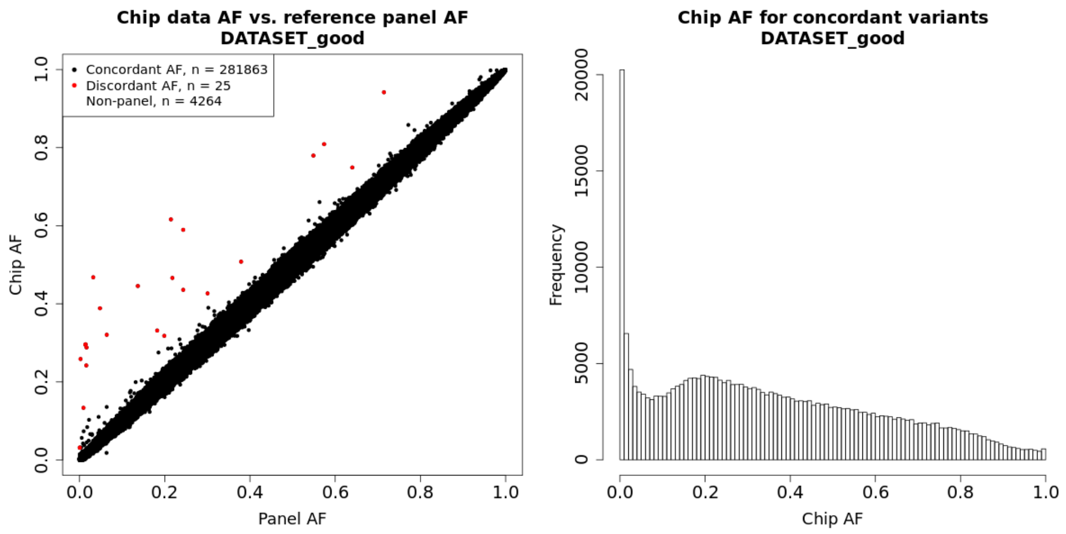
Command

```
DATASET=your_dataset_prefix
PANEL_FRQ=/path/to/panel.frq

# Copy and save the given 'plot_AF.R' file and run it with:
Rscript --no-save /path/to/plot_AF.R \
    ${DATASET}_chip.frq \
    ${DATASET} \
    ${PANEL_FRQ} \
    0.1 \
    5
```

Command

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

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

# Input variables
args <- commandArgs(TRUE)
infrq <- args[1] # Full path to chip data .frq file
indataset <- args[2] # Dataset name prefix
sub_dataset <- basename(indataset) # Subset the dataset name
ref_panel <- args[3] # Full path to reference panel .frq file
af_diff <- as.numeric(args[4]) # Max AF difference to panel
af_fc <- as.numeric(args[5]) # Max AF fold change to panel

# Read in the frequency files
chip <- fread(infrq, header = TRUE)
panel <- fread(ref_panel, header = TRUE)

# Take an intersection of the panel and chip data
# based on SNP column (in format CHR_POS_REF_ALT)
isec <- merge(panel, chip, by = c("CHR", "SNP", "REF", "ALT"))
colnames(isec)[c(5,6)] <- c("AF_PANEL", "AF_CHIP")

isec$AF_diff <- abs(isec$AF_PANEL - isec$AF_CHIP)
isec$AF_ratio <- isec$AF_CHIP/isec$AF_PANEL
isec$AF_FC <- log2(isec$AF_ratio)

# Check if AFs are within the pp and fold change ranges
af_ok <- ((isec$AF_diff < af_diff) &
         ((isec$AF_FC < af_fc) & (isec$AF_FC > -af_fc)))

exclude <- !af_ok

# Generate an exclusion list for variants not in the panel
nonpanel <- chip[!(chip$SNP) %in% (isec$SNP)]
# Generate AF list for discordant variants
af_discrepant <- isec[exclude]

# Save the plots as png
png(paste0(sub_dataset, "_AFs.png"),
    width = 1200, height = 600)
    par(mfrow = c(1,2))
    par(cex.axis = 1.6, cex.lab = 1.5, cex.main = 1.6)

    # Plot first all and then excludable variants
    plot(isec$AF_PANEL, isec$AF_CHIP, col=1, pch=20,
           main=paste0("Chip data AF vs. reference panel AF\n",
           sub_dataset),
           xlab="Panel AF", ylab="Chip AF")
    points(isec[exclude]$AF_PANEL, isec[exclude]$AF_CHIP,
            col=2, pch=20)
    # Draw a legend
    legend("topleft", legend=c(
           paste0("Concordant AF, n = ", nrow(isec[!exclude])),
           paste0("Discordant AF, n = ", nrow(isec[exclude])),
           paste0("Non-panel, n = ", nrow(nonpanel))),
           col=c("black", "red", "white"), pch=20, cex=1.2)

    # Add chip AF histogram to the same png
    hist(isec[!exclude]$AF_CHIP, breaks=100,
            main=paste0("Chip AF for concordant variants\n",
            sub_dataset),
            xlab="Chip AF")
dev.off()

# Write out the exclusion lists (SNPID for bcftools)
# Highly discrepant AF values
write.table(isec[exclude]$SNP,
        paste0(sub_dataset, "_exclude.txt"),
        quote=F, row.names=F, col.names=F)
# Variants noit present in the reference panel
write.table(nonpanel$SNP,
        paste0(sub_dataset, "_nonpanel_exclude.txt"),
        quote=F, row.names=F, col.names=F)

# Write out the isec without exclusions
write.table(isec[!exclude]$SNP,
        paste0(sub_dataset, "_panel_isec.txt"),
        quote=F, row.names=F, col.names=F)

# Write out the discrepant AFs in case needed
write.table(af_discrepant,
        paste0(sub_dataset, "_discrepant_AF.txt"),
        quote = F, row.names = F, sep = "\t")
```

Expected result



Example of a good dataset plots with a population-specific imputation reference panel.

8 Exclude the variants showing highly discordant allele frequencies or not present in the imputation reference panel as listed in the previous step.

Input files:

- DATASET_exclude.txt (**Step7**)
- DATASET_nonpanel_exclude.txt (**Step 7**)
- DATASET_AF.vcf.gz (**Step 5**)

Outpt files:

- DATASET_exclusion1.txt
- DATASET_exclusion2.txt
- DATASET_for_phasing.vcf.gz

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

Command

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

```
DATASET=your_dataset_prefix

# Sort the exclusion lists
sort -V ${DATASET}_exclude.txt \
    > ${DATASET}_exclusion1.txt
sort -V ${DATASET}_nonpanel_exclude.txt \
    > ${DATASET}_exclusion2.txt

# Exclude the variants showing highly discrepant AF
bcftools view -e ID=@${DATASET}_exclusion1.txt  \
    ${DATASET}_AF.vcf.gz \
    -Oz -o ${DATASET}_exclusion1.vcf.gz

# Exclude the variants not present in the reference panel
bcftools view -e ID=@${DATASET}_exclusion2.txt \
    ${DATASET}_discrepant_AF_excluded.vcf.gz \
    -Oz -o ${DATASET}_for_phasing.vcf.gz
```

Chip data pre-phasing

9 Chip data pre-phasing will speed up the genotype imputation step.
Phase haplotypes for each chromosome separately before imputation.

Here, adjust the number of threads and parallelization of the phasing for each chromosomes according to your computing environment. We generally use 8 threads per chromosome and run all chromosomes in parallel distributed on 23 cores.

Input files:
- DATASET_for_phasing.vcf.gz (**Step 8**)
- eagle_chr#_b38.map (where # is chromosome number) (**Step 1.2.2**)

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.

Command

```
DATASET=your_dataset_prefix

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

    # Chromosome 23 is coded as 'chrX',
    # whereas autosomal chrs are 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
```

Genotype imputation

10 Run genotype imputation for each chromosome separately.

Here, adjust the number of threads and parallelization of the imputation for each chromosomes according to your computing environment. We have observed the fastest runs by using 16 threads pre chromosome and running all chromosomes in parallel distributed on 23 cores.

*NOTE: Chromosome X should be represented as diploid genotypes. If this is not the case, fix the ploidy with the example command from **Step 1.3.3**.*

Input files:

- DATASET_for_imputation_chr#.vcf.gz (**Step 9**)
- panel_phased_AF_chr#.bref (where # is chromosome number) (**Step 1.3.6**)
- beagle_chr#_b38.map (where # is chromosome number) (**Step 1.2.3**)

Outpt file:

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

Command

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

```
DATASET=your_dataset_prefix
BEAGLE_PATH=/path/to/beagle
REFERENCE_BREF=/path/to/panel_phased_AF

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

    java -Xss5m -Xmx16g \
      -jar ${BEAGLE_PATH}/beagle.27Jan18.7e1.jar \
        gt=${DATASET}_for_imputation_chr${CHR}.vcf.gz \
        ref=${REFERENCE_BREF}_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
```

Post-imputation processing and quality assurance

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

Here, we use bcftools plugin +fill-tags to only add AF value. However, the plugin is able to add also other values, such as Hardy-Weinberg equilibrium p-value, if required. See documentation for details at: http://samtools.github.io/bcftools/howtos/plugin.fill-tags.html

Input file:
- DATASET_imputed_chr#.vcf.gz (where # is chromosome number) (**Step 10**)

Outpt files:
- DATASET_imputed_chr#.vcf.gz (where # is chromosome number)
- DATASET_imputed_info_chr#.vcf.gz.tbi (where # is chromosome number)

Command

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 <dataset>_imputed_chr${chr}.vcf.gz -Ou

```
DATASET=your_dataset_prefix

# 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
```

12 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) (**Step 11**)

Outpt file:
- DATASET_INFO_group_chr#.txt (where # is chromosome number)

Command

```
DATASET=your_dataset_prefix

# Generate an allele frequency file for plotting for each chrs
for CHR in {1..23}; do

    # Generate a header for the output file
    echo -e 'CHR\tSNP\tREF\tALT\tAF\tINFO\tAF_GROUP' \
        > ${DATASET}_INFO_group_chr${CHR}.txt

    # Query the required fields and
    # add frequency group (1, 2 or 3) as the last 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 | \
    # Here $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 }' \
        >> ${DATASET}_INFO_group_chr${CHR}.txt
done
```

13 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 imputation reference panel.

Copy and store the R script below 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 below.

Input files:

- DATASET_INFO_group_chr#.txt (where # is chromosome number) (**Step 12**)
- panel.frq (**Step 1.3.5**)

Outpt files:

- DATASET_chr#_postimputation_summary_plots.png (where # is chromosome number)
- DATASET_postimputation_summary_plots.pdf

!! 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.

Command

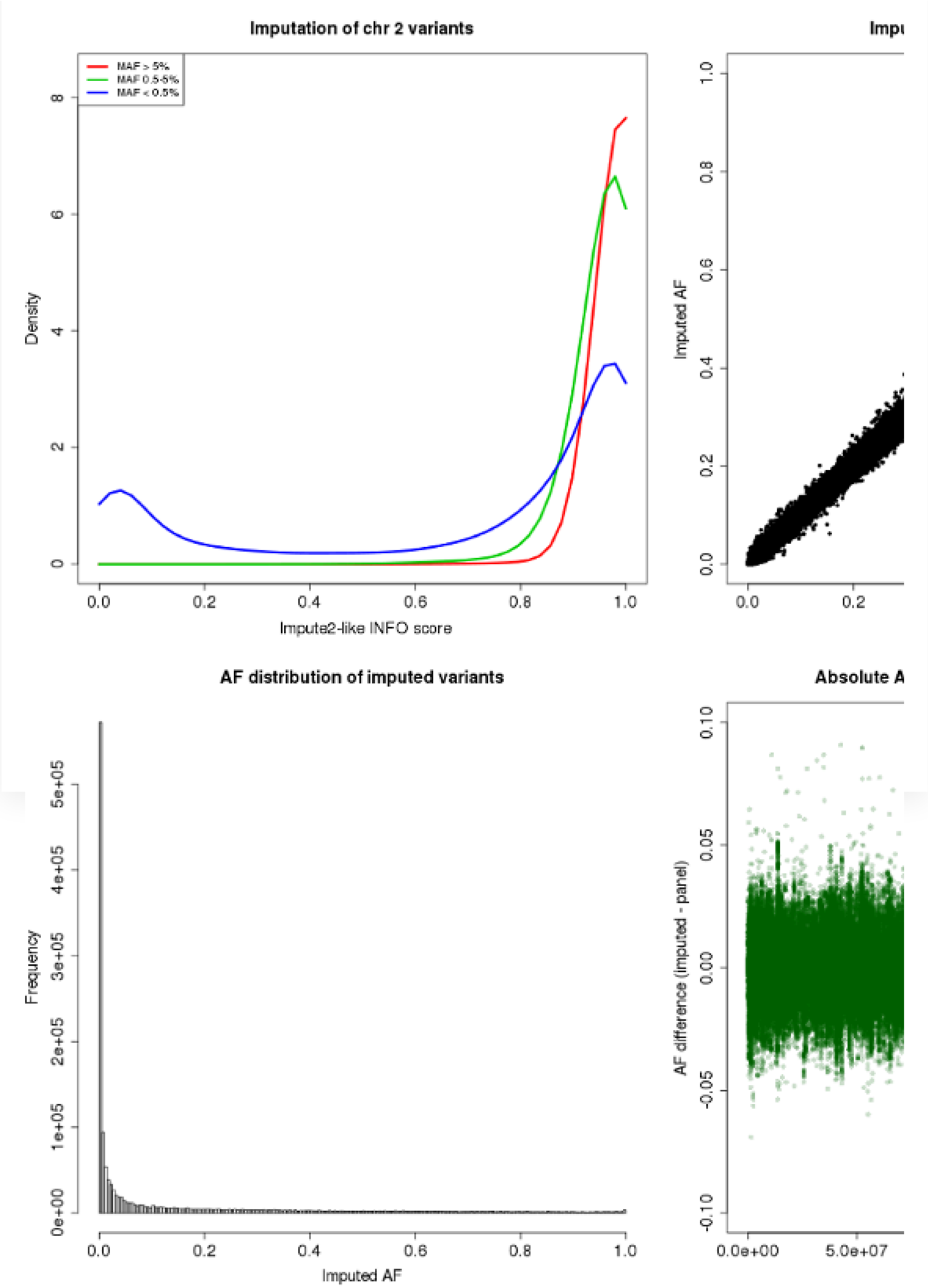
```
DATASET=your_dataset_prefix
DATASET_INFO=${DATASET}_INFO_group
OUTPUT=${DATASET}_postimputation_summary_plots

# Copy and save the given 'plot_INFO_and_AF_for_imputed_chrs.R' file
# and run it with:
for CHR in {1..23}; do
    Rscript --no-save  \
        plot_INFO_and_AF_for_imputed_chrs.R \
        ${DATASET_INFO}_chr${CHR}.txt \
        ${OUTPUT}_chr${CHR} \
        ${REFERENCE_FRQ}
done

# Combine the plots per chromosome into a single pdf file
convert $(ls ${OUTPUT}_chr*.png | sort -V) \
     ${OUTPUT}.pdf
```

Command

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

# Required libraries
library(data.table) # For fast fread
library(sm)         # For density plotting

# Input variables
args <- commandArgs(TRUE)
infile <- args[1]       # Input file
outdataset <- args[2]    # Output prefix
panel.frq <- args[3]    # Path to panel frq file
panel <- fread(panel.frq, header = TRUE)

# Creates the filename as DATASET_tag_chr#.png
png(filename = paste0(outdataset,  ".png"),
    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_INFO_group_chr#.txt
    imp_vars <- fread(infile), header = TRUE)

    # Create random sample for INFO-value plot
    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-value distributions
    sm.density.compare(temp1$INFO, temp1$AF_GROUP,
        xlab = "INFO-value", ylab = "Density",
        col = col, lty = rep(1,3), lwd = 3,
        xlim = c(0,1), cex = 1.4, h = 0.05)
    title(paste0("Imputation of chr ", chr, " variants"))
    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 panel 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),'_',fixed=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, maxColorValue=255),
        ylim = c(-0.1, 0.1), pch = 16,
        xlab = "Chromosome position",
        ylab = "AF difference (imputed - panel)")

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

Expected result

Example of chromosome 2 post-imputation summary plots with a population-specific imputation reference panel.


Imputation of chr 2 variants
MAF > 5%
MAF 0.5-5%
MAF < 0.5%
Density
0
2
4
6
8
0.0
0.2
0.4
0.6
0.8
1.0
Impute2-like INFO score
Impu
Imputed AF
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
AF distribution of imputed variants
Frequency
0e+00
1e+05
2e+05
3e+05
4e+05
5e+05
0.0
0.2
0.4
0.6
0.8
1.0
Imputed AF
Absolute A
AF difference (imputed - panel)
-0.10
-0.05
0.00
0.05
0.10
0.0e+00
5.0e+07