Protocol: what after Snps array vcf?

Step1: cleaning and organizing vcf of axiom software

1- Adjust the header info HWp

Command1

```
sed 's/OLDWORD/NEWWORD/g' input.vcf > output.vcf
```

2- Add extra ID to info header in vcf

Command 1

```
cat > extra_info_header.hdr <<EOF
##INFO=<ID=chr_id,Number=1,Type=String,Description="Chromosome ID from array annotation">
##INFO=<ID=start,Number=1,Type=String,Description="Start position from array annotation">
##INFO=<ID=stop,Number=1,Type=String,Description="Stop position from array annotation">
##INFO=<ID=strand,Number=1,Type=String,Description="Strand information">
##INFO=<ID=dbsnp_loctype,Number=1,Type=String,Description="dbSNP location type">
##INFO=<ID=in_hapmap,Number=1,Type=String,Description="Present in HapMap">
##INFO=<ID=strand_vs_dbsnp,Number=1,Type=String,Description="Strand compared to dbSNP">
##INFO=<ID=probe_count,Number=1,Type=String,Description="Number of probes">
##INFO=<ID=cytoband,Number=1,Type=String,Description="Cytoband">
##INFO=<ID=chrx_par,Number=1,Type=String,Description="X chromosome pseudoautosomal region flag">
##INFO=<ID=flank,Number=1,Type=String,Description="Flanking sequence">
##INFO=<ID=allele_a,Number=1,Type=String,Description="Allele A from array">
##INFO=<ID=allele_b,Number=1,Type=String,Description="Allele B from array">
##INFO=<ID=ref_allele,Number=1,Type=String,Description="Reference allele from annotation">
##INFO=<ID=alt_allele,Number=1,Type=String,Description="Alternative allele from annotation">
##INFO=<ID=associated_gene,Number=1,Type=String,Description="Associated gene symbol">
##INFO=<ID=genetic_map,Number=1,Type=String,Description="Genetic map position">
##INFO=<ID=microsatellite,Number=1,Type=String,Description="Microsatellite flag/info">
##INFO=<ID=heterozygous_allele_frequencies,Number=.,Type=String,Description="Heterozygous allele frequencies">
##INFO=<ID=allele_frequency_count,Number=.,Type=String,Description="Allele frequency counts">
##INFO=<ID=allele_frequencies,Number=.,Type=String,Description="Allele frequencies">
##INFO=<ID=minor_allele,Number=1,Type=String,Description="Minor allele">
##INFO=<ID=minor_allele_frequency,Number=1,Type=String,Description="Minor allele frequency">
##INFO=<ID=omim,Number=.,Type=String,Description="OMIM annotation">
##INFO=<ID=biomedical,Number=.,Type=String,Description="Biomedical annotation">
##INFO=<ID=annotation_notes,Number=.,Type=String,Description="Annotation notes">
##INFO=<ID=ordered_alleles,Number=.,Type=String,Description="Ordered alleles">
##INFO=<ID=allele_count,Number=.,Type=String,Description="Allele count">
EOF
```

Command 2

```
bcftools annotate -h extra_info_header.hdr  -Oz -o vcf_with_info_header.vcf.gz input.vcf
tadix vcf_with_info_header.vcf.gz
```

Filters applied in the article:

- Autosomal only
- SNPs only (A/C/G/T)

- Bi-allelic only
- MAF > 5%
- Variant missingness < 10%
- Sample missingness < 5%
- HWE p > 1e−6
- Remove ambiguous SNPs (AT/CG)
- Remove mismatched allele frequencies
- LD pruning: $r^2 < 0.1$, 500 kb
- Remove sex mismatches
- Remove heterozygosity outliers
- Remove ancestry outliers via PCA
- Remove related individuals (>0.0884)

Step1b: Create VCF available for plink

```
bcftools annotate \
  -x INFO \
  -Oz -o vcf_noinfo.vcf.gz \
  vcf_with_info_header.vcf.gz

tabix  vcf_noinfo.vcf.gz
```

Step2: applying the filter list to vcf_noinfo.vcf.gz

```
awk 'BEGIN{OFS="\t"} \
    NR==1 {print "#FID","IID","SEX"; next} \
    {sex = ($2=="female" ? 2 : ($2=="male" ? 1 : 0));
     print $1, $1, sex}' sex_raw.txt > sex_info.txt
```

1- Convert your SNP-array VCF to PLINK format (autosomal, bi-allelic SNPs)

Command

```
singularity exec plink_combo.sif plink2
--vcf vcf_noinfo.vcf.gz
--snps-only just-acgt
--max-alleles 2
 --make-pgen
--out snps_array_qc0
--allow-extra-chr
 --threads 32
--split-par hg38
 --update-sex sex_info.txt
```

2- Extract QC statistics (with FILTERS)

🟦 SAMPLE-LEVEL QC

1 autosome

```
singularity exec plink_combo.sif plink2 \
  --pfile snps_array_step1_nosex  --autosome  --make-pgen  --out snps_array_step1 --allow-extra-chr
```

2 Remove samples with high missingness (> 5%)

```
singularity exec plink_combo.sif plink2 \
  --pfile snps_array_step1  --mind 0.05  --make-pgen  --out snps_array_step2_mind
```

3 Remove heterozygosity outliers

```
singularity exec plink_combo.sif plink2 \
 --pfile snps_array_step2_mind \
 --het \
 --out het
```

Detect outliers in R (|F − mean(F)| > 3SD), produce:

```
# Read het.het (do NOT treat '#' as comment)
het <- read.table("het.het",
            header = TRUE,
            comment.char = "",
            stringsAsFactors = FALSE)

# Fix column name: X.IID -> IID
names(het)[1] <- "IID"

# Check
print(colnames(het))
# Should be: "IID" "O.HOM." "E.HOM." "OBS_CT" "F"

# Mean and SD of F
meanF <- mean(het$F, na.rm = TRUE)
sdF   <- sd(het$F,   na.rm = TRUE)

# Flag outliers: |F - mean(F)| > 3 SD
het$outlier <- abs(het$F - meanF) > 3 * sdF

# Subset outliers, keep as data.frame (drop = FALSE!)
out <- het[het$outlier, c("IID"), drop = FALSE]

if (nrow(out) > 0) {
 out2 <- data.frame(FID = out$IID, IID = out$IID)
 write.table(out2, "het_outliers.txt",
          col.names = FALSE,
          row.names  = FALSE,
          quote     = FALSE)
} else {
 # no outliers -> create empty file
 file.create("het_outliers.txt")
}
```

het_outliers.txt

```
singularity exec plink_combo.sif plink2 \
 --pfile snps_array_step2_mind \
 --remove het_outliers.txt \
 --make-pgen \
 --out snps_array_step3_nohet
```

4 **Remove related individuals (>0.0884)**

```
singularity exec plink_combo.sif plink2 \
 --pfile snps_array_step3_nohet \
 --king-cutoff 0.0884 \
 --make-pgen \
 --out snps_array_step4_unrelated
```

SNP-LEVEL QC

6 **Filter SNPs: missingness <10%, MAF>5%, HWE>1e−6**

```
singularity exec plink_combo.sif plink2 \
 --pfile snps_array_step4_unrelated \
 --geno 0.10 \
 --maf 0.05 \
 --hwe 1e-6 \
 --make-pgen \
 --out snps_array_step6_snpqc
```

7 Remove ambiguous SNPs (A/T, T/A, C/G, G/C)

```
awk 'NR>1 && (($4=="A" && $5=="T") ||
  ($4=="T" && $5=="A") ||
  ($4=="C" && $5=="G") ||
  ($4=="G" && $5=="C")) {print $3}' snps_array_step6_snpqc.pvar > ambiguous_snps.txt
```

```
singularity exec plink_combo.sif plink2 \
  --pfile snps_array_step6_snpqc \
  --exclude ambiguous_snps.txt \
  --make-pgen \
  --out snps_array_step7_noambig
```

8 Remove mismatched allele frequencies

(*only if you have reference GWAS frequencies*)

```
plink2 \
  --pfile snps_array_step7_noambig \
  --freq cols=chrom,ID,pos,ref,alt,altfreq \
  --out snps_array_step7_noambig_freq
```

Using bcftools (recommended, simple):

1 Make sure AF is present (or add it)

```
bcftools +fill-tags turk_wgs.mgvcf.gz \
  -Oz -o turk_wgs_AF.vcf.gz \
  -- -t AF,AC,AN
```

2 Extract a clean frequency table

```
bcftools query \
  -f '%CHROM\t%POS\t%ID\t%REF\t%ALT\t%INFO/AF\n' \
  turk_wgs_AF.vcf.gz \
  > turk_wgs_freq.txt
(echo -e "CHROM\tPOS\tID\tREF\tALT\tAF_ref"; cat turk_wgs_freq.txt) > turk_wgs_freq_header.txt
```

2 Merge in R and find mismatched SNPs

```
## 1. Read target (array) frequencies
target <- read.table("snps_array_step7_noambig_freq.afreq", header = TRUE)
## Sometimes first column is named '#CHROM' → rename to 'CHROM'
names(target)[1] <- "CHROM"
## 2. Read Turkish WGS reference frequencies
ref <- read.table("turk_wgs_freq_header.txt", header = TRUE)
## 3. Merge by CHROM, POS, REF, ALT
merged <- merge(
  target,
  ref,
  by = c("CHROM", "POS", "REF", "ALT")
)
## 4. Compute absolute difference |freq_target - freq_ref|
## target AF column = ALT_FREQS
## ref AF column = AF_ref
merged$diff <- abs(merged$ALT_FREQS - merged$AF_ref)
## 5. Flag SNPs with big mismatch, e.g. > 0.15
bad <- merged[merged$diff > 0.15, "ID"]
## 6. Write list of mismatched SNP IDs for PLINK
write.table(
  bad,
  "freq_mismatch_snps.txt",
  col.names = FALSE,
  row.names = FALSE,
  quote = FALSE
```

)

3 Exclude mismatched SNPs in PLINK2

```
plink2 \
  --pfile snps_array_step7_noambig \
  --exclude freq_mismatch_snps.txt \
  --make-pgen \
  --out snps_array_step8_nomismatch
```

Calculate PRS:
Get it from gwas website gwas_raw.tsv
Such as https://ftp.ebi.ac.uk/pub/databases/gwas/summary_statistics/GCST90654001-GCST90655000/

```
awk 'NR==1 {print "SNP\tA1\tBETA"; next} \
    (length($4)==1 && length($5)==1 && $6!="NA") { \
        print $2 ":" $3, toupper($4), $6 \
    }' Trans-ethnic_GWAS_meta-analysis_without_replication.txt \
  > weights_clean.txt
```

```
singularity exec plink_combo.sif plink2   --pfile snps_array_step7_noambig   --set-all-var-ids @:#   --make-pgen   --out snps_array_step7_chrpos   --threads 32
```

```
singularity exec plink_combo.sif plink2   --pfile snps_array_step7_chrpos   --score weights_clean.txt 1 2 3 header-read cols=+scoresums   --out prs_score   --threads 32
```

or

```
# USER SETTINGS

# Singularity image with plink2 + plink1.9

SIF="plink_combo.sif"

# BASE DATA: GWAS summary stats (the file you showed)

BASE_GWAS="Trans-ethnic_GWAS_meta-analysis_without_replication.txt"

# TARGET DATA: your QC'd Turkish SNP-array pfile (already chr:pos IDs)

TARGET_PREFIX="snps_array_step7_chrpos"

# P-value thresholds for C+T

PRS P_THRESH_LIST="5e-8 1e-6 1e-4 1e-3 1e-2 5e-2 1e-1 5e-1 1"

OUTDIR="prs_results"

mkdir -p "${OUTDIR}"
```

1. PREPARE CLEAN BASE GWAS

This whole block takes your raw GWAS summary stats, filters them, standardises SNP IDs to CHR:BP, and creates a clean weight file (SNP, effect allele, beta) that PLINK can use to calculate PRS.

```
# BASE_GWAS FORMAT:

# RSID CHR BP_hg19 A1 A2 BETA SE P

echo "[STEP 1] Cleaning base GWAS summary stats..."

CLEAN_SUMSTATS="${OUTDIR}/base_sumstats_clean.txt"

WEIGHTS_ALL="${OUTDIR}/weights_all.txt"

awk '
  NR==1 {
      print "SNP\tCHR\tBP\tA1\tA2\tBETA\tP";
      next
  }
  (tolower($4) ~ /^[acgt]$/ &&
   tolower($5) ~ /^[acgt]$/ &&
   $6 != "NA" &&
   $8 != "NA") {

      # SNP ID as CHR:BP (e.g. 10:100000625)
      printf "%s:%s\t%s\t%s\t%s\t%s\t%s\t%s\n",
            $2, $3,      # CHR:BP  (for SNP ID)
            $2,          # CHR
            $3,          # BP
            toupper($4), # A1
            toupper($5), # A2
            $6,          # BETA
            $8           # P
  }' "${BASE_GWAS}" > "${CLEAN_SUMSTATS}"

# PRS weights file (SNP, A1, BETA) used later by –score

awk '
  NR==1 {print "SNP\tA1\tBETA"; next}
  {print $1, $5, $6}
' OFS="\t" "${CLEAN_SUMSTATS}" > "${WEIGHTS_ALL}"

echo "[STEP 1] Clean sumstats -> ${CLEAN_SUMSTATS}" echo "[STEP 1] Weights file -> ${WEIGHTS_ALL}"
```

2. MAKE BED FILE FOR CLUMPING

This step converts your pfile to PLINK1 bed format and stores its name in BED_PREFIX so you can use it in the clumping step.

```
# (plink1.9 --clump needs bed/bim/fam)

echo "[STEP 2] Converting pfile to bed for clumping..."

singularity exec "${SIF}" plink2 \
--pfile "${TARGET_PREFIX}" \
--make-bed \
--out "${OUTDIR}/${TARGET_PREFIX}_bed" \
--threads 32

BED_PREFIX="${OUTDIR}/${TARGET_PREFIX}_bed"
```

3. LD CLUMPING (LD adjustment)

This step uses your genotypes to do LD clumping on the GWAS summary stats, then outputs a list of independent lead SNPs and their p-values that you'll use to build PRS at different p-value thresholds.

```
echo "[STEP 3] Running LD clumping (r2=0.1, 250kb)..."
CLUMP_OUT="${OUTDIR}/base_clump"

# Use plink1.9 here
singularity exec "${SIF}" plink \
--bfile "${BED_PREFIX}" \
--clump "${CLEAN_SUMSTATS}" \
--clump-p1 1 \
--clump-p2 1 \
--clump-r2 0.1 \
--clump-kb 250 \
--out "${CLUMP_OUT}"

# Extract the clumped index SNPs list with their p-values

CLUMPED_SNPS="${OUTDIR}/clumped_snps_with_p.txt"

awk 'NR>1 {print $3"\t"$5}' "${CLUMP_OUT}.clumped" > "${CLUMPED_SNPS}"

# columns: SNP P

echo "[STEP 3] Clumped SNPs -> ${CLUMPED_SNPS}"
```

4. BUILD SCORE FILES FOR MULTIPLE P THRESHOLDS

STEP 4 loops over each p-value threshold, selects the clumped lead SNPs with P ≤ that threshold, and builds a separate PRS weight file (SNP, A1, BETA) for each threshold so you can calculate different PRS models and see which threshold predicts best.

```
echo "[STEP 4] Building PRS weight files for each P threshold..."

for PTHR in ${P_THRESH_LIST}; do
  TAG=$(echo "${PTHR}" | sed 's/e-*/e/g; s/\./p/g; s/-//g')  # e.g. 5e-8 -> 5e8, 0.05 -> 0p05
  SNP_LIST="${OUTDIR}/snps_p${TAG}.txt"
  SCORE_FILE="${OUTDIR}/weights_p${TAG}.txt"

  # 4a. get SNPs from clumped list with P <= threshold
  awk -v thr="${PTHR}" '($2+0) <= thr {print $1}' "${CLUMPED_SNPS}" > "${SNP_LIST}"

  N_SNPS=$(wc -l < "${SNP_LIST}" || echo 0)
  echo "  P <= ${PTHR} : ${N_SNPS} SNPs"

  if [ "${N_SNPS}" -eq 0 ]; then
    echo "    [WARN] No SNPs at this threshold, skipping."
    continue
  fi

  # 4b. make PRS weight file (SNP A1 BETA) for those SNPs
  awk 'NR==FNR {keep[$1]=1; next}
       NR==1   {print; next}
       ($1 in keep)' "${SNP_LIST}" "${WEIGHTS_ALL}" > "${SCORE_FILE}"

done
```

5. CALCULATE PRS IN TARGET DATA

This step loops over each p-value threshold, and for each one uses PLINK2 --score with your weight file to calculate a PRS (SCORE1_SUM) for every individual, saving the results into .sscore files.

```
echo "[STEP 5] Calculating PRS for each P-value threshold..."

for PTHR in ${P_THRESH_LIST}; do
```

```
  TAG=$(echo "${PTHR}" | sed 's/e-*/e/g; s/\./p/g; s/-//g')
  SCORE_FILE="${OUTDIR}/weights_p${TAG}.txt"
  if [ ! -s "${SCORE_FILE}" ]; then
    echo "  [SKIP] No score file for P <= ${PTHR}"
    continue
  fi

  OUT_PREFIX="${OUTDIR}/prs_p${TAG}"
  singularity exec "${SIF}" plink2 \
    --pfile "${TARGET_PREFIX}" \
    --score "${SCORE_FILE}" 1 2 3 header-read cols=+scoresums \
    --out "${OUT_PREFIX}" \
    --threads 32

  echo "  [DONE] PRS at P <= ${PTHR} -> ${OUT_PREFIX}.sscore"
done

echo "[ALL DONE] PRS pipeline finished. Check ${OUTDIR}/*.sscore"
```

We constructed clumping-and-thresholding PRS following standard guidelines. Briefly, GWAS summary statistics were LD-clumped using PLINK (r^2=0.1, 250 kb window), and PRS were generated across multiple p-value thresholds (5×10^{-8}–1). For each threshold we evaluated prediction performance (incremental R^2 / AUC) in the target dataset, and selected the optimal threshold based on maximal performance. Model stability was assessed using K-fold cross-validation and, where available, an independent validation cohort, consistent with recommendations in existing PRS protocols.