한국인칩v2 정도관리 프로토콜 (v1.0)

2024. 08.

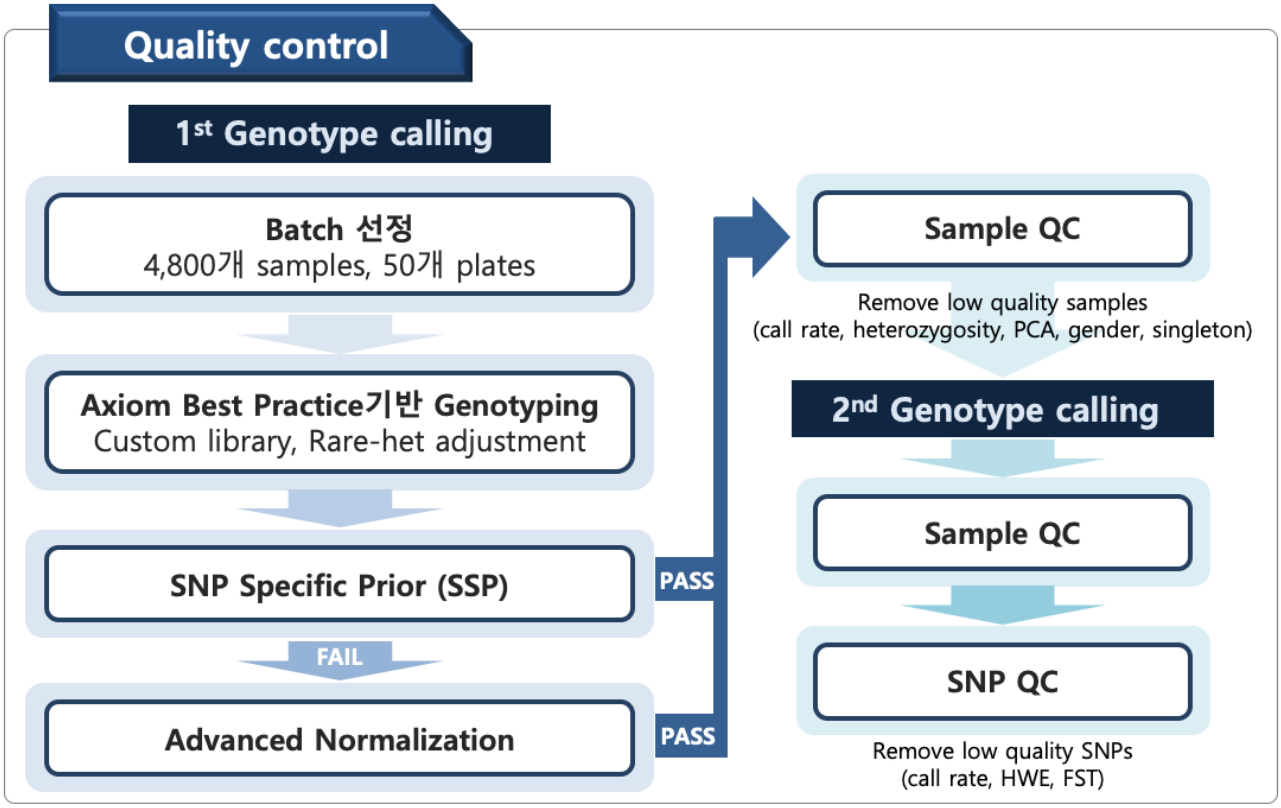
질병관리청
국립보건연구원
유전체연구기술개발과

목 차

한국인칩* 유전체정보 정도관리 개요

* 공개 중인 한국인칩 유전체정보는 v2.0A과 v2.0B로 생산 되었으며, 정도관리 프로토콜은 한국인칩 v2.0B로 생산된 경우에만 해당함 (한국인칩 v2.0A는 '한국인칩v1_정도관리프로토콜' 참고)

1. 정도관리 흐름도



〈한국인칩 정도관리 흐름도〉

2. 준비

1) 프로그램

· Affymetrix Power Tool(APT)[1] 설치
· simple-ssp[1] 설치
· advnorm.sh[1] 설치
· PLINK[2] 설치
· R 통계 프로그램[3] 설치

---

1) APT 설치와 설행은 관련 홈페이지 참고 (https://www.thermofisher.com)
2) PLINK 설치와 실행은 관련 홈페이지 참고 (http://www.cog-genomics.org/plink)
3) R 설치와 실행은 관련 홈페이지 참고 (http://www.r-project.org)

· Python 프로그램[4] 설치
· FlashPCA 프로그램[5] 설치
· KING 프로그램[6] 설치

2) 라이브러리

· Axiom_KBAbetaB_r2_3

3) 데이터

· 원시 데이터(CEL 파일)

4) 제공 파일

· SSP 모델 (KBAv2.0_SSP_v1_202406.models)
· KBAv2_quality_control_input_v1_202408.txt

4) Python 설치와 실행은 관련 홈페이지 참고 (https://www.python.org/downloads/)
5) FlashPCA 설치와 실행은 관련 홈페이지 참고 (https://github.com/gabraham/flashpca)
6) KING 설치와 실행은 관련 홈페이지 참고 (https://www.kingrelatedness.com/history.html)

1차 Genotype calling

1. 1차 Genotype calling

1) 배치 선정

○ 내용

· 플레이트(plate) 50개로 구성된 샘플 4,800개 이하로 배치 선정

* Plate: 96개 샘플을 한번에 genotyping 하는 단위

2) Genotype calling

○ 내용

· Axiom BestPractice 기반 각 배치 genotype calling 수행

○ 명령어의 예

```
apt-genotype-axiom \
 --analysis-files-path library_path \
 --arg-file library_path/argument_file.xml \
 --out-dir output_path \
 --cel-files cel_file_list.txt \
 --log-file apt-genotype-axiom.log \
 --do-rare-het-adjustment True \
 --summaries \
 --genotyping-node:snp-posteriors-output true \
 --artifact-reduction-output-trustcheck true
```

○ 옵션

· --analysis-files-path: 라이브러리 경로

· --arg-file: 라이브러리 안에 있는 argument_file.xml 파일명

· --out-dir: 결과 파일이 저장될 경로

· --cel-files: CEL파일 리스트 파일명 (파일의 header는 반드시 cel_files)

· --log-file: log파일명

· --do-rare-het-adjustment: rare variant calling algorithm 사용

· --summaries: A와 B allele에 대한 intensity 정보가 있는 파일 생성

· --genotyping-node:snp-posteriors-output true: 각 프로브 당 genotype cluster의 location과 variance 정보가 있는 파일 생성

· --artifact-reduction-output-trustchecktrue: 오류 가능성이 있는 프로브 정보가 있는 파일 생성

3) SNPolisher 분석

○ 내용

· QC matrix 생성을 통해 SNPolisher 분석

○ 명령어의 예

```
ps-metrics \
 --posterior-file AxiomGT1.snp-posteriors.txt \
 --multi-posterior-file AxiomGT1.snp-posteriors.multi.txt \
 --call-file AxiomGT1.calls.txt \
 --summary-file AxiomGT1.summary.txt \
 --metrics-file metrics.txt \
 --log-file ps-metrics.log
ps-classification \
 --species-type human \
 --metrics-file ./output/metrics.txt \
 --multi-metrics-file ./outut/multi-metrics.txt \
 --output-dir . \
 --log-file ps-classification.log
```

4) 결과파일 생성

○ 내용

· 정도관리를 위한 VCF 파일 생성

○ 명령어의 예

```
apt-format-result \
 --calls-file AxiomGT1.calls.txt \
 --annotation-file library_path/.annot.db \
 --export-vcf-file KBAv2.0B_Batch*.vcf \
 --log-file apt-format-result.log
bgzip -f KBAv2.0B_Batch*.vcf
```

○ 옵션

· --calls-file: '2) Genotype calling' 과정에서 생성된 AxiomGT1.calls.txt 파일명
· --annotation-file: 라이브러리 안에 있는 annot.db 파일명
· --export-vcf-file: 결과 파일명
· --log-file: log파일명

○ 내용
· multi-allele normalization 및 left-align
○ 파일 준비
· header.txt, rename_chr.txt, GRCh38.primary_assembly.genome.fa (파일 제공)
○ 명령어의 예

```
# GRCh38 버전에 맞게 염색체 변경
bcftools annotate --header-lines header.txt KBAv2.0B_Batch*.vcf.gz | \
bcftools annotate --rename-chrs rename_chr.txt | \
bgzip -c > KBAv2.0B_Batch*_rechr.vcf.gz

# multi-allele normalization
bcftools norm -m -any KBAv2.0B_Batch*_rechr.vcf.gz | \
bgzip -c > KBAv2.0B_Batch*_rechr_norm.vcf.gz

# left-align
bcftools norm -f GRCh38.primary_assembly.genome.fa -c x
KBAv2.0B_Batch*_rechr_norm.vcf.gz | \
bcftools annotate --set-id '%ID|%CHROM:%POS:%REF:%ALT' | \
bgzip -c > KBAv2.0B_Batch*_rechr_norm_annoprb_lftaln.vcf.gz
```

○ 옵션
· annotate —header-lines: 추가할 해더를 포함하고 있는 파일명
· annotate —rename-chr: 변경할 염색체를 포함하고 있는 파일명
· norm -m -any: multi-allele normalization
· norm -f .fa -c x: .fa 파일과 reference가 같지 않으면 삭제하고, left-align

○ 내용

· 정도관리를 위한 PLINK 파일(fam, bim, bed) 생성

○ 명령어의 예

```
plink \
 --vcf KBAv2.0B_Batch*_rechr_norm_annoprb_lftaln.vcf.gz \
 --double-id \
 --allow-extra-chr \
 --make-bed \
 --out KBAv2.0B_Batch*
```

○ 옵션

· --vcf: vcf 파일명

· --double-id: vcf 샘플 아이디로 FID, IID 생성

· --allow-extra-chr: 비표준염색체 포함

· --make-bed: 결과파일 PLINK(fam, bim, bed) 형식

· --out: 결과 파일 preifx

2. 1차 SNP Specific Prior(SSP)를 적용한 genotype calling

1) SSP 모델 input 생성

* 3개 이상의 batch (약 1.5만 명 이상)이 있을 경우 수행. 그 외에는 2-3)에서 미리 생성하여 제공하는 SSP 파일을 활용

○ 내용

· SSP 모델 생성을 위한 performance.txt, posterior.txt 파일 생성

· ConversionType이 PolyHighResolution, MonoHighResolution, NoMinorResolution, Hemizygous인 SNP 중 call rate 값이 가장 높은 batch의 SNP 정보를 사용하여 생성한 모델

○ 명령어의 예

```
python3
import pandas as pd

batch_num = n  # n은 배치 개수
for i in range(1, batch_num + 1):
    file_path1 = f'Ps.performance_Batch{i}.txt'
    globals()[f'batch{i}'] = pd.read_csv(file_path1, comment = '#', sep = '\t')
```

```
    globals()[f'g_batch{i}'] = globals()[f'batch{i}'].loc[:, ['probeset_id',
'ConversionType', 'CR']]
    globals()[f'g_batch{i}'].rename(columns={'ConversionType':
f'ConversionType{i}', 'CR': f'CR{i}'}, inplace=True)

g_m = g_batch1
for i in range(2, batch_num + 1):
    g_m = pd.merge(g_m, globals()[f'g_batch{i}'], on = 'probeset_id')

poly_columns = ['ConversionType' + str(i) for i in range(1, batch_num + 1)]
vpass = ['PolyHighResolution', 'NoMinorHom', 'MonoHighResolution',
'Hemizygous']
vbool = (g_m[poly_columns].isin(vpass)).sum(axis = 1) != 0
df1 = g_m[vbool]

for i in vpass:
    df1[i] = (df1[poly_columns] == i).sum(axis=1)

war1 = df1.loc[(df1['PolyHighResolution'] != 0) &
(df1['MonoHighResolution'] != 0) & (df1['NoMinorHom'] != 0),:]

war2 = df1.loc[(df1['PolyHighResolution'] != 0) &
(df1['MonoHighResolution'] != 0) & (df1['NoMinorHom'] == 0),:]

war_id = pd.concat([war1['probeset_id'], war2['probeset_id']])
df2 = df1[~(df1['probeset_id'].isin(war_id))]
df2 = df2.reset_index(drop = True)

vtypes = []
for i in range(0, len(df2)):
    if (df2['Hemizygous'][i] != 0) :
        vtypes.append('Hemizygous')
    elif (df2['PolyHighResolution'][i] != 0) :
        vtypes.append('PolyHighResolution')
    elif ((df2['PolyHighResolution'][i] == 0) & (df2['NoMinorHom'][i] != 0)) :
        vtypes.append('NoMinorHom')
    else: vtypes.append('MonoHighResolution')
```

```
df2['vtype'] = vtypes
type_columns = ['Type' + str(i) for i in range(1, batch_num + 1)]

df2['vbatch'] = ''
for index, row in df2.iterrows():
    vtype2 = row['vtype']
    max_cr = -1
    max_batch = ''

    for i in range(1, batch_num + 1):
        current_type = row[f'ConversionType{i}']
        current_cr = row[f'CR{i}']

        if current_type == vtype2 and current_cr > max_cr:
            max_cr = current_cr
            max_batch = str(i)

    df2.at[index, 'vbatch'] = max_batch

for i in range(1, batch_num + 1) :
     globals()[f'b{i}'] = df2.loc[df2['vbatch'] == str(i), 'probeset_id']
     globals()[f'b{i}_per'] =
globals()[f'batch{i}'][globals()[f'batch{i}']['probeset_id'].isin(list(globals()[f'b{i}']))]

b_per = ['b' + str(i) + '_per' for i in range(1, batch_num + 1)]
merge_performance = b1_per
for vdf in b_per[1:]:
    vper = globals()[vdf]
    merge_performance = pd.concat([merge_performance, vper])

for i in range(1, batch_num + 1) :
    file_name = f'AxiomGT1.snp-posteriors_Batch{i}.txt'
    globals()[f'pos_{i}'] = pd.read_csv(file_name, comment = '#', sep = '\t')
    globals()[f'b{i}_pos'] =
globals()[f'pos_{i}'][(globals()[f'pos_{i}']['id'].isin(list(globals()[f'b{i}'])) |
(globals()[f'pos_{i}']['id'].isin(globals()[f'b{i}'] + ':1'))]
```

```
b_pos = ['b' + str(i) + '_pos' for i in range(1, batch_num + 1)]
merge_poster = b1_pos
for vdf in b_pos[1:]:
    vpos = globals()[vdf]
    merge_poster = pd.concat([merge_poster, vpos])

merge_performance.to_csv('merge_performance.txt', index = False,
header = True, sep = '\t', na_rep = 'NA')
merge_poster.to_csv('merge_posterior.txt', index = False, header =
True, sep = '\t', na_rep = 'NA')
```

2) SSP 모델 생성

○ 내용

· SSP 모델 생성

○ 명령어의 예

```
sh simple-ssp \
 --posterior-file merge_posteriors.txt \
 --performance-file merge_performance.txt \
 --output-dir output_path \
 --log-file ssp_log.txt
```

○ 옵션

· --posterior-file: ‘1) SSP 모델 input 생성’ 과정에서 생성된 posterior 파일명

· --performance-file: ‘1) SSP 모델 input 생성’ 과정에서 생성된 performance 파일명

· --output-dir: 결과 파일이 저장될 경로

· --log-file: log파일명

3) Genotype calling (SSP 모델 사용)

○ 내용

· SSP 모델로 prior adjustment 통한 genotype calling

○ 파일 준비

· SSP model 파일 (KBAv2.0B_SSP_v1_202408.models 제공)

○ 명령어의 예

```
apt-genotype-axiom \
 --analysis-files-path library_path \
 --arg-file library_path/argument_file.xml \
 --out-dir output_path \
 --cel-files cel_file_list.txt \
 --genotyping-node:snp-prior-input-file
KBAv2.0B_SSP_v1_202408.models \
 --log-file apt-genotype-axiom.log \
 --do-rare-het-adjustment True \
 --summaries \
 --genotyping-node:snp-posteriors-output true \
 --artifact-reduction-output-trustcheck true
```

○ 옵션

· --genotyping-node:snp-prior-input-file: ‘2) SSP 모델 생성’ 과정에서 생성된 모델

4) SNPolisher 분석

○ 내용

· ‘1. Genotype calling 3) SNPolisher 분석’ 과정과 동일

5) 결과파일 생성

○ 내용

· ‘1. Genotype calling 4) 결과파일 생성’ 과정과 동일

3. 1차 Advanced normalization (AdvNorm) 적용한 genotype calling

1) Genotype calling, SNPolisher 분석

○ 내용

· Intensity adjustment 통한 genotype calling

○ 명령어의 예

```
advnorm.sh \
  --summary-file AxiomGT1.summary.txt \
  --calls-file AxiomGT1.calls.txt \
  --report-file AxiomGT1.report.txt \
  --trustcheck-file AxiomGT1.trustcheck.txt \
  --analysis-files-path library_path \
  --snp-specific-param-file library_path/.snp_specific_analysis.txt \
  --snp-priors-file KBAv2.0B_SSP_v1_202408.models \
  --special-snps-file library_path/.specialSNPs \
  --ps2snp-file library_path/.ps2snp_map.ps \
  --output-dir output_path
```

○ 옵션

· --summary-file: '2. 1차 SNP SSP를 적용한 genotype calling' 과정에서 생성된 AxiomGT1.summary.txt 파일명

· --calls-file: '2. 1차 SNP SSP를 적용한 genotype calling' 과정에서 생성된 AxiomGT1.calls.txt 파일명

· --report-file: '2. 1차 SNP SSP를 적용한 genotype calling' 과정에서 생성된 AxiomGT1.report.txt 파일명

· --trustcheck-file: '2. 1차 SNP SSP를 적용한 genotype calling' 과정에서 생성된 AxiomGT1.trustcheck.txt 파일명

· --analysis-files-path: 라이브러리 경로

· --snp-specific-param-file: 라이브러리 안에 있는 snp_specific_analysis.txt 파일명

· --snp-priors-file: '2. 1차 SNP SSP를 적용한 genotype calling' 과정에서 생성된 모델명

· --special-snps-file: 라이브러리 안에 있는 .specialSNPs 파일명

· --ps2snp-file: 라이브러리 안에 있는 .ps2snp_map.ps 파일명

· --output-dir: 결과 파일이 저장될 경로

2) 결과파일 생성

○ 내용

· '1. Genotype calling 4) 결과파일 생성' 과정과 동일

1차 High quality SNP 선별

1. SNPolisher 분석을 통한 High quality SNP 선별

○ 내용

· 1. Genotyping, 2. SSP, 3. AdvNorm SNPolisher 분석결과로 high quality SNP 선별

○ 파일 준비

· 1. Genotyping, 2. SSP, 3. AdvNorm 과정의 SNPolisher 분석결과 Ps.performance.txt

○ 기준

· ConversionType이 PolyHighResolution, MonoHighResolution, NoMinorHom, Hemizygous 인 SNP 추출

○ 명령어의 예

```
python3
import pandas as pd

genotyping = pd.read_csv('1stGENO/Ps.performance.txt', comment='#',
sep='\t', usecols=[0, 14])
ssp = pd.read_csv('1stSSP/Ps.performance.txt', comment='#', sep='\t',
usecols=[0, 14])
adv = pd.read_csv('1stADV/Ps.performance.txt', comment='#', sep='\t',
usecols=[0, 16])

for df_name, i in [('geno', genotyping), ('ssp', ssp), ('adv', adv)]:
  vbool1 = []
  for j in i['ConversionType']:
    if j in ['PolyHighResolution', 'MonoHighResolution', 'NoMinorHom',
'Hemizygous']:
      vbool1.append(True)
    else:
      vbool1.append(False)
  i[f'vbool_{df_name}'] = vbool1

merge1 = pd.merge(genotyping, ssp, on = 'probeset_id')
merge2 = pd.merge(merge1, adv, how = 'left', on = 'probeset_id')
```

```
genotyping_pass = merge2.loc[merge2['vbool_geno']==True, 'probeset_id']
ssp_pass = merge2.loc[((merge2['vbool_geno']==False)
                       & (merge2['vbool_ssp']==True)), 'probeset_id']
adv_pass = merge2.loc[((merge2['vbool_geno']==False)
                       & (merge2['vbool_ssp']==False)
                       & (merge2['vbool_adv']==True)), 'probeset_id']

genotyping_pass.to_csv('1stGENO/Batch*_1stpass.txt', header=False, index=False)
ssp_pass.to_csv(1stSSP/Batch*_1stpass.txt', header=False, index=False)
adv_pass.to_csv('1stADV/Batch*_1stpass.txt', header=False, index=False)
```

2. High quality SNP 추출

○ 내용

· 1. Genotyping, 2. SSP, 3. AdvNorm 각 과정별 선별된 high quality SNP 추출

○ 명령어의 예

```
plink \
 --bfile KBAv2.0B_Batch* \
 --extract Batch*_1stpass.txt \
 --allow-extra-chr \
 --make-bed \
 --out KBAv2.0B_Batch*_1stpass
```

○ 옵션

· --bfile: 1. Genotyping, 2. SSP, 3. AdvNorm genotype calling 과정에서 생성된 PLINK 파일 preifx

· --extract: '1. SNPolisher 분석을 통한 High quality SNP 선별' 과정에서 생성된 파일명

3. 데이터 통합

○ 내용

· 1. Genotyping, 2. SSP, 3. AdvNorm 각 과정별 추출된 high quality SNP 통합

○ 명령어의 예

```
plink \
 --merge-list mergelist.txt \
 --allow-extra-chr \
 --make-bed \
 --out KBAv2.0B_Batch*_1stmerge
```

○ 옵션

· --merge-list: 1. Genotyping, 2. SSP, 3. AdvNorm 각 과정별 선별된 high quality SNP 추출 결과로 생성된 bed, bim, fam 파일 절대경로가 포함된 파일명 (구분자: 탭)

* mergelist.txt 예

```
1stGENO/KBAv2.0B_Batch*_1stpass.bed 1stGENO/KBAv2.0B_Batch*_1stpass.bim
1stGENO/KBAv2.0B_Batch*_1stpass.fam
1stSSP/KBAv2.0B_Batch*_1stpass.bed 1stSSP/KBAv2.0B_Batch*_1stpass.bim
1stSSP/KBAv2.0B_Batch*_1stpass.fam
1stADV/KBAv2.0B_Batch*_1stpass.bed 1stADV/KBAv2.0B_Batch*_1stpass.bim
1stADV/KBAv2.0B_Batch*_1stpass.fam
```

1차 샘플 정도관리

1. 저품질 샘플 제거

1) Low call rate

○ 내용

· Call rate가 낮은 샘플은 DNA quality가 낮거나 실험상의 오류로 인해 발생할 수 있으므로 제거

○ 명령어의 예

```
plink \
  --bfile KBAv2.0B_Batch*_1stmerge \
  --missing \
  --allow-extra-chr \
  --out KBAv2.0B_Batch*_1stmerge_CR
```

○ 옵션

· --missing: missing rate 계산

2) Excessive heterozygosity

○ 내용

· Heterozygosity가 높은 경우 DNA quality가 낮거나 실험 중 샘플 contamination에 발생할 가능성이 있으므로 제거

○ 명령어의 예

```
plink \
  --bfile KBAv2.0B_Batch*_1stmerge \
  --het \
  --allow-extra-chr \
  --out KBAv2.0B_Batch*_1stmerge_HET
```

○ 옵션

· --het: 각 샘플 별로 homo, hetero genotype count를 측정

3) 저품질 샘플 제거

○ 기준

· 전체 샘플 분포를 확인하여 전체 샘플 분포에서 크게 벗어나는 경우 제거 (‘1) Low call rate’, ‘2) Excessive heterozygosity’ 과정에서 생산된 결과 사용)

- ‘F_MISS > 0.05 | HET < 20.5 | 24.5 <HET’인 경우의 샘플 제거

○ 도식화의 예

```
R
miss <- read.table("KBAv2.0B_Batch*_1stmerge_CR.imiss", header=T)
het <- read.table("KBAv2.0B_Batch*_1stmerge_HET.het", header=T)
miss <- cbind(miss, CR=((1-miss$F_MISS)*100))
het <- cbind(het, HET=((het$N.NM. - het$O.HOM.)/het$N.NM.)*100)
data <- merge(miss, het, by="FID")
pdf("CR_HET_plot.pdf")
plot(data$het, data$F_MISS)
dev.off()
q()
```

<저품질 샘플 분포 확인의 예>

○ 명령어의 예

```
plink \
 --bfile KBAv2.0B_Batch*_1stmerge \
 --remove rmCRHET.txt \
 --allow-extra-chr \
 --make-bed \
 --out KBAv2.0B_Batch*_1stmerge_rmCRHET
```

○ 옵션

· --remove: ‘1) Low call rate’, ‘2) Excessive heterozygosity’ 과정에서 선별된 샘플의 FID, IID 정보를 포함하는 파일명 (구분자: 탭)

2. SNP pruning & Principal Conponent Analysis (PCA)

1) SNP pruning

○ 내용

· PCA 분석은 전체 데이터를 사용할 경우 매우 많은 시간이 소요되기 때문에 데이터에서 LD를 기반으로 대표 SNP 정보만 선별

○ 명령어의 예

```
plink \
 --bfile KBAv2.0B_Batch*_1stmerge_rmCRHET \
 --maf 0.1 \
 --geno 0.01 \
 --hwe 0.001 \
 --indep-pairwise 50 5 0.5 \
 --not-chr 6,14 \
 --allow-extra-chr \
 --out KBAv2.0B_Batch*_1stmerge_prune
```

○ 옵션

· --maf: Minor Allele Frequency(MAF) 기준으로 제시된 수치 미만의 SNP 제거

· --geno: SNP의 missing rate(1-call rate)가 제시된 수치를 초과한 SNP 제거

· --hwe: Hardy Weinberg Equilibrium을 만족하지 않는 SNP($P < 10^{-3}$) 제거

· --indep-pairwise [한번에 계산할 SNP 수] [다음 계산을 위해 건너뛰는 SNP 수] [LD r2 기준]: 맨 처음 SNP부터 50개씩 LD r2를 계산하여 대표되는 SNP을 선별하고 동일한 계산을 5개의 SNP씩 옮겨서 다음의 50개 SNP에 대해 계산

· --not-chr: 6,14번 염색체 제외

* chr 6, 14의 경우 특정 지역(MHC 등)에 SNP 수가 많아 PCA 분석 시 bias된 결과를 발생시킴

2) Pruned SNP 추출

○ 명령어의 예

```
plink \
 --bfile KBAv2.0B_Batch*_1stmerge_rmCRHET\
 --extract KBAv2.0B_Batch*_1stmerge_prune.prune.in \
 --allow-extra-chr \
 --make-bed \
 --out KBAv2.0B_Batch*_1stmerge_rmCRHET_prune
```

3) PCA

○ 내용

· PCA 결과 전체 샘플의 cluster에서 벗어나있는 경우 genetic ancestry가 다르거나,

population stratification, artifact의 가능성이 있어 분석에서 제거

○ 명령어의 예

```
flashpca \
 --bfile KBAv2.0B_Batch*_1stmerge_rmCRHET_prune \
 --outpc KBAv2.0B_Batch*_1stmerge_rmCRHET_prune_PCA.txt
```

○ 옵션

· --bfile: Pruned SNP 추출된 PLINK prefix

· --outpc: PCA 결과 파일명

○ 도식화의 예

```
R
PCA <-
read.table(file="KBAv2.0B_Batch*_1stmerge_rmCRHET_prune_PCA.txt",
header=T)
pdf("PCA_plot.pdf")
plot(PCA$PC1, PCA$PC2)
dev.off()
q()
```

〈 PCA 분석 결과 도식화 및 제거 샘플 확인 〉

○ 명령어의 예

```
plink \
 --bfile KBAv2.0B_Batch*_1stmerge_rmCRHET \
 --remove rmPCA_batch*.txt \
 --allow-extra-chr \
 --make-bed \
 --out KBAv2.0B_Batch*_1stmerge_rmCRHET_rmPCA
```

○ 옵션

· --remove: '3) PCA' 과정에서 선별된 샘플의 FID, IID 정보를 포함하는 파일명 (구분자: 탭)

2차 Genotype calling & High quality SNP 선별

1. 2차 genotype calling

○ 내용

· 1차 샘플 정도관리 항목 중 저품질 샘플들을 제거 후 genotype calling 수행 (‘1차 Genotype calling’ 과정과 동일)

○ 기준

· Call rate & Heterozygosity 기준: rmCRHET.txt

· PCA outlier 기준: rmPCA.txt

2. 2차 High quality SNP 선별

○ 내용

· ‘1차 High quality SNP 선별’ 과정과 동일

2차 샘플 정도관리

1. 2차 샘플 정도관리

○ 내용

· '1차 샘플 정도관리' 과정을 수행하여 샘플 정도관리 기준에 만족하지 못하는 샘플 제거

2. Relationship inference

1) 모든 배치 통합

○ 내용

· '1. 2차 샘플 정도관리' 과정에서 생성된 결과 파일 (KBAv2.0B_Batch*_2ndSampleQC) 통합

○ 명령어의 예

```
plink \
 --merge-list mergelist.txt \
 --allow-extra-chr \
 --make-bed \
 --out KBAv2.0B_2ndSampleQC_merge
```

○ 옵션

· --merge-list: '1. 2차 샘플 정도관리' 과정에서 생성된 bed, bim, fam 파일 절대경로가 포함된 파일명 (구분자: 탭)

2) KING 분석

○ 내용

· 샘플 사이의 관계를 통계 분석하여 동일인이라고 추정되는 샘플 리스트 생성

○ 명령어의 예

```
king \
 -b KBAv2.0B_2ndSampleQC_merge.bed \
 --duplicate \
 --prefix KBAv2.0B_2ndSampleQC_merge_KING

sed 1d KBAv2.0B_2ndSampleQC_merge_DUP.con | cut -f1,3 | sed
's/\t/\n/g' | awk '{print $1"\t"$1}' > rmKING.txt
```

○ 옵션

· -b: ‘1) 모든 배치 통합’ 과정에서 생성된 bed 파일명

· --duplicate: 중복된 샘플만 결과 파일 생성

· --prifix: 결과 파일 prefix

3) 동일인 샘플 제거

○ 내용

· 동일인이라고 추정되는 샘플 제거

○ 명령어의 예

```
plink \
  --bfile KBAv2.0B_Batch*_2ndSamapleQC \
  --remove rmKING.txt \
  --allow-extra-chr \
  --make-bed \
  --out KBAv2.0B_Batch*_2ndSampleQC_rmKING
```

○ 옵션

· --remove: king 결과에서 추출한 동일인 샘플의 FID, IID 정보를 포함하는 파일명 (구분자: 탭)

3. 성별 정보 확인

1) 성별 확인

○ 내용

· X, Y 염색체 비율을 계산하여 샘플의 성별 추측

○ 파일 준비

· updateSEX.txt: 샘플의 FID, IID와 성별정보(남성=1, 여성=2, missing=0) 포함한 파일 (구분자: 탭)

○ updateSEX.txt 예

```
Sample1    Sample1    0
Sample2    Sample2    1
Sample3    Sample3    2
```

○ 명령어의 예

```
plink \
--bfile KBAv2.0B_Batch*_2ndSampleQC_rmKING \
--update-sex updateSEX.txt \
--allow-extra-chr \
--make-bed \
--out KBAv2.0B_Batch*_2ndSampleQC_rmKING_updateSEX

plink \
 --bfile KBAv2.0B_Batch*_2ndSampleQC_rmKING_updateSEX \
 --check-sex \
 --allow-extra-chr \
 --out KBAv2.0B_Batch*_2ndSampleQC_rmKING_checkSEX
```

○ 옵션

· --check-sex: 성별 확인

2) 성별 불일치 샘플 제거

○ 내용

· 성별 불일치 샘플을 제거

○ 기준

· 남자(F < 0.8), 여자(F > 0.2) 제거

· 단, 별도의 성별확인 실험을 통해 정확한 성별이 확인된 경우 F 값이 기준에 만족하지 못해도 제거하지 않음

○ 명령어의 예

```
grep PROBLEM
KBAv2.0B_Batch*_2ndSampleQC_rmKING_checkSEX.sexcheck | awk
'{print $1"\t"$2}' > rmSEX.txt

plink \
 --bfile KBAv2.0B_Batch*_2ndSampleQC_rmKING \
 --remove rmSEX.txt \
 --allow-extra-chr \
 --make-bed \
 --out KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX
```

○ 옵션

· --remove: 성별 불일치하는 샘플의 FID, IID 정보를 포함하는 파일명 (구분자: 탭)

SNP 정도관리

1. 중복 SNP 리스트 제거

○ 내용

· 정도관리 전 control probe 등 중복아이디를 가지는 SNP 제거

○ 파일 준비

· 중복 SNP ID 파일 (KBAv2_quality_control_input_v1_202408.txt 제공)

○ 명령어의 예

```
grep KBAv2.0B KBAv2_quality_control_input_v1_202408.txt | awk
'$7=="REMOVE" {print $2"|chr"$3}'
KBAv2_quality_control_input_v1_202408.txt > rmDUP.txt

plink \
 --bfile KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX \
 --extract rmDUP.txt \
 --allow-extra-chr \
 --make-bed \
 --out KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX_rmDUP
```

○ 옵션

· --extract: 중복 SNP ID 정보를 포함하는 파일명

2. Autosome 및 Pseudoautosomal Region(PAR) 정도관리

○ 내용

· 상염색체(1~22번)에 포함되는 autosomal 및 PAR 지역관련 SNP을 추출하여 정도관리 수행

1) Autosome 및 PAR SNP 추출

○ 내용

· Autosomal 및 PAR 지역 관련 SNP 추출

○ 명령어의 예

```
plink \
  --bfile KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX_rmDUP \
  --autosome-xy \
  --allow-extra-chr \
  --make-bed \
  --out KBAv2.0B_Batch*_Autosome
```

○ 옵션

· --autosome-xy: 상염색체와 PAR 지역 관련 SNP 추출

* 주의: bim 파일에 있는 염색체25번(PAR)가 있을 경우 --merge-x를 이용해서 24번으로 변환 후 활용해야함

2) Low call rate SNP 제거

○ 내용

· Call rate가 낮은 SNP는 probe design, clustering 분석 문제 등이 있어 calling 과정에서 오류가 발생할 수 있으므로 제거

○ 기준

· Call rate가 95% 미만인 경우 제거

○ 명령어의 예

```
plink \
  --bfile KBAv2.0B_Batch*_Autosome \
  --geno 0.05 \
  --allow-extra-chr \
  --make-bed \
  --out KBAv2.0B_Batch*_Autosome_rmGENO
```

○ 옵션

· --geno: missing rate 기준 이상 제거

3) Hardy Weinberg Equilibrium(HWE) 기준 SNP 제거

○ 내용

· HWE test P-value가 매우 낮을 경우 genotype clustering 과정 등에서 오류가 발생하였을 가능성이 있으므로 분석에서 제거

○ 기준

· HWE P-value가 10-6 미만인 경우 제거

○ 명령어의 예

```
plink \
 --bfile KBAv2.0B_Batch*_Autosome_rmGENO \
 --hwe 1e-6 \
 --allow-extra-chr \
 --make-bed \
 --out KBAv2.0B_Batch*_Autosome_rmGENO_rmHWE
```

○ 옵션

· --hwe: p-value 기준 미만 제거

4) F-statistics(Fst) 분석

○ 내용

· 유전변이의 빈도가 한국인, 아시아인에서 나타나는 빈도와 많은 차이가 날 경우 genotype clustering 오류 등의 문제가 있을 가능성이 있으므로 분석에서 제거

· 단, 질환의 특성을 고려하여 적용여부를 결정

4-1) Minor Allele Frequency(MAF) 값 구하기

○ 명령어의 예

```
plink2 \
 --bfile KBAv2.0B_Batch*_Autosome_rmGENO_rmHWE
 --freq \
 --allow-extra-chr \
 --make-bed \
 --out KBAv2.0B_Batch*_maf
```

4-2) Fst 계산

○ 파일 준비

· 한국인, 아시아인 빈도 정도가 있는 파일 (KBAv2_quality_control_input_v1_202408.txt 제공)

· 배치별 minor allele frequency 계산된 파일 (KBAv2.0B_Batch*_maf.afreq)

○ 명령어의 예

```
python 3
import pandas as pd
```

```
wgs = pd.read_csv('KBAv2_quality_control_input_v1_202408.txt', sep='\t',
usecols = ['ProbeID', 'WGS8K_MAF', 'gnomAD_EAS_MAF', 'FST'])

vmaf = []
for i in range(0, len(wgs)):
    if (wgs['WGS8K_MAF'][i] == '-') :
        vmaf.append(wgs['gnomAD_EAS_MAF'][i])
    else : vmaf.append(wgs['WGS8K_MAF'][i])
wgs['wgs_MAF'] = vmaf

wgs2 = wgs[wgs['FST'] == 'FST']
wgs2['wgs_MAF'] = wgs2['wgs_MAF'].astype('float')
wgs2 = wgs2.reset_index(drop = True)
rm = wgs.loc[wgs['FST'] == 'REMOVE', 'SNP']
data = pd.read_csv('KBAv2.0B_Batch*_maf.frq', sep='\s+', header=0,
usecols=[1,4])
vmerge = pd.merge(wgs2, data, on = 'SNP')
vmerge['mean_frq'] = (vmerge['wgs_MAF'] + vmerge['MAF']) / 2
vmerge['var_freq'] = ((vmerge['wgs_MAF'] - vmerge['mean_frq'])**2 +
(vmerge['MAF'] - vmerge['mean_frq'])**2) / 2
vmerge['fst'] = vmerge['var_freq'] / (vmerge['mean_frq'] * (1 -
vmerge['mean_frq']))

output = vmerge.loc[(vmerge['fst'] > 0.025), 'SNP']
output2 = rm[rm.isin(data['SNP'])]
vresult = pd.concat([output, output2])
vresult.to_csv('KBAv2.0B_Batch*_rmFST.txt', header = False, index =
False)
```

4-3) 빈도 차이가 나는 SNP 제거

○ 기준

· Fst 값이 0.025 이상 차이가 나는 경우 제거

* threshold 값은 유전변이 정확도 등을 고려하여 선정됨

○ 명령어의 예

○ 옵션

```
plink \
 --bfile KBAv2.0B_Batch*_Autosome_rmGENO_rmHWE \
 --exclude KBAv2.0B_Batch*_rmFST.txt \
 --allow-extra-chr \
 --make-bed \
 --out KBAv2.0B_Batch*_Autosome_rmGENO_rmHWE_rmFST
```

· --exclude: Fst 값이 0.025 이상 차이가 나는 SNP ID 정보를 포함하는 파일명

5) 모든 배치 데이터 통합 (선택 사항 : Batch가 2개 이상인 경우 진행)

○ 내용

· 배치별 SNP 정도관리를 통과한 autosome 데이터 통합

○ 명령어의 예

```
plink \
 --merge-list mergelist.txt \
 --allow-extra-chr \
 --make-bed \
 --out KBAv2.0B_Autosome_merge
```

○ 옵션

· --merge-list: SNP QC를 통과한 autosome 변이 bed, bim, fam 파일 절대경로가 포함된 파일명 (구분자: 탭)

6) 통합 후 SNP 정도관리 (선택 사항 : Batch가 2개 이상인 경우 진행)

○ 내용

· 통합 후, call rate가 낮은 SNP 및 HWE test P-value가 매우 낮은 SNP 제거

○ 기준

· Call rate가 90% 미만인 경우 제거

· HWE P-value가 10^{-6} 미만인 경우 제거

○ 명령어의 예

```
plink \
 --bfile KBAv2.0B_Autosome_merge \
 --hwe 1e-6 \
 --geno 0.1 \
 --allow-extra-chr \
 --make-bed \
 --out KBAv2.0B_Autosome_merge_SNPQC
```

○ 옵션

· --geno: missing rate 기준 이상 제거

· --hwe: p-value 기준 미만 제거

3. Non-autosome SNP 정도관리

○ 내용

· 염색체 23번(X) non-PAR 지역 및 염색체 24번(Y)와 26번(미토콘드리아) SNP 추출하여 non-autosomal 정도관리 수행

1) 염색체 23번(X) SNP 정도관리

○ 내용

· 염색체 23번은 남자와 여자를 분리하여 SNP 정도관리를 수행하고 통합 후 SNP 정도관리 수행

○ 명령어의 예

```
plink \
 --bfile KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX_rmDUP \
 --split-x hg38 \
 --allow-extra-chr \
 --make-bed \
 --out KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX_rmDUP_splitX

plink \
 --bfile KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX_rmDUP_splitX
\
 --chr 23 \
 --allow-extra-chr \
 --make-bed \
```

```
 --out
KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX_rmDUP_splitX_CHR23

plink \
 --bfile
KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX_rmDUP_splitX_CHR23 \
 --filter-males \
 --set-hh-missing \
 --allow-extra-chr \
 --make-bed \
 --out
KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX_rmDUP_splitX_CHR23_MALE

plink \
 --bfile
KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX_rmDUP_splitX_CHR23 \
 --filter-females \
 --hwe 1e-6 \
 --allow-extra-chr \
 --make-bed \
 --out
KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX_rmDUP_splitX_CHR23_FEMALE

plink \
 --bfile
KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX_rmDUP_splitX_CHR23_MALE \
 --bmerge
KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX_rmDUP_splitX_CHR23_FEMALE \
 --allow-extra-chr \
 --make-bed \
 --out
KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX_rmDUP_splitX_CHR23_MERGE

plink \
 --bfile
KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX_rmDUP_splitX_CHR23_MERGE \
```

```
  --geno 0.05 \
  --allow-extra-chr \
  --make-bed \
  --out
KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX_rmDUP_splitX_CHR23_SNPQC
```

○ 옵션

· --split-x: 염색체 25번이 존재하지 않는 경우 23번, 25번으로 염색체 분리

· --chr: 지정한 염색체만을 선택

· --filter-males: 남성만을 선택

· --set-hh-missing: heterozygous를 missing으로 처리

· --filter-females: 여성만을 선택

2) 염색체 24번(Y), 26번(미토콘드리아) SNP 추출

○ 내용

· 염색체 24번, 26번 SNP 추출

○ 명령어의 예

```
plink \
  --bfile KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX_rmDUP \
  --chr 24,26 \
  --allow-extra-chr \
  --make-bed \
  --out
KBAv2.0B_Batch*_2ndSampleQC_rmKING_rmSEX_rmDUP_CHR24_26
```

3) 염색체 23번(X), 24번(Y), 26번(미토콘드리아) 통합

○ 내용

· SNP 정도관리가 완료된 non-autosome 데이터 통합

○ 명령어의 예

```
plink \
  --merge-list merge.list \
  --allow-extra-chr \
  --make-bed \
  --out KBAv2.0B_Batch*_NonAutosome
```

○ 옵션

· --merge-list: SNP 정도관리가 완료된 non-autosome bed, bim, fam 파일 절대경로가 포함된 파일명 (구분자: 탭)

4) 모든 배치 데이터 통합 (선택 사항 : Batch가 2개 이상인 경우 진행)

○ 내용

· 모든 배치 SNP 정도관리가 완료된 non-autosome 데이터 통합

○ 명령어의 예

```
plink \
 --merge-list mergelist.txt \
 --allow-extra-chr \
 --make-bed \
 --out KBAv2.0B_NonAutosome_merge
```

○ 옵션

· --merge-list: 각 배치 SNP 정도관리가 완료된 non-autosome bed, bim, fam 파일 절대경로가 포함된 파일명 (구분자: 탭)

4. Autosome, Non-autosome 데이터 최종 통합

○ 내용

· SNP 정도관리가 완료된 autosome, non-autosome 데이터 통합

○ 명령어의 예

```
plink \
 --bfile KBAv2.0B_Autosome_merge_SNPQC \
 --bmerge KBAv2.0B_NonAutosome_merge \
 --allow-extra-chr \
 --make-bed \
 --out KBAv2.0B_finalSNPQC
```

5. SNP ID 변환 (필요시 수행)

○ 내용

· Probe ID로 되어있는 SNP ID를 chr:pos:ref:alt(hg38) 형식으로 변환

○ 파일 준비

· Probe ID와 chr:pos:ref:alt(hg38) 정보가 담긴 파일

(KBAv2_quality_control_input_v1_202408.txt 제공)

○ 명령어의 예

```
awk '{$2"|chr"$3"\t"$3}' KBAv2_quality_control_input_v1_202408.txt > KBAv2.0B_updateSNP.txt

plink \
 --bfile KBAv2.0B_finalSNPQC \
 --update-name KBAv2.0B_updateSNP.txt \
 --allow-extra-chr \
 --make-bed \
 --out KBAv2.0B_finalSNPQC_updateSNP
```

한국인칩 통합 분석

○ 내용

· 동일한 샘플을 한국인칩 v2.0A와 v2.0B를 생산 했을 경우, 두 버전에 공통으로 존재하는 SNP 제거, 정도관리 후 통합

· Autosome은 'SNP 정도관리 2. Autosome 및 PAR 정도관리 4-3) 빈도 차이가 나는 SNP 제거'까지 수행한 후, non-Autosome은 'SNP 정도관리 3. Non-autosome SNP 정도관리 3) 염색체 23번, 24번, 26번 통합'한 후 수행

1. 공통으로 존재하는 SNP 제거

○ 내용

· 동일한 SNP인데 Probe ID가 다른 경우 KBAv1.1 SNP ID로 통일

○ 명령어의 예

```
awk '$7=="INTERSNP" {print $2"|chr"$3}'
KBAv2_quality_control_input_v1_202408.txt > rmINTERSNP.txt

# Autosome
plink \
 --bfile KBAv2.0B_Batch*_Autosome_rmGENO_rmHWE_rmFST \
 --exclude rmINTERSNP.txt \
 --make-bed \
 --out KBAv2.0B_Batch*_Autosome_rmGENO_rmHWE_rmFST_rmINTERSNP

# Non-autosome
plink \
 --bfile KBAv2.0B_Batch*_NonAutosome \
 --update-name updateNAME.txt \
 --make-bed \
 --out KBAv2.0B_Batch*_NonAutosome_rmINTERSNP
```

2. 칩간 reference 버전 통일 (필요시)

○ 내용

· hg38에서 hg19로 변환

○ 파일 준비

· updateCHRMAP.txt: 마커 아이디별 hg19 염색체, 위치 정보가 담긴 파일 (구분자:탭)

· updateALLELE.txt: 마커 아이디별 reference allele, alternative allele 정보가 담긴 파일 (구분자:탭)

○ 명령어의 예

```
 # Autosome
plink \
 --bfile KBAv2.0B_Batch*_Autosome_rmGENO_rmHWE_rmFST_rmINTERSNP \
 --update-chr updateCHRMAP.txt 2 1 \
 --update-map updateCHRMAP.txt 3 1 \
 --update-alleles updateALLELE.txt \
 --make-bed \
 --out
KBAv2.0B_Batch*_Autosome_rmGENO_rmHWE_rmFST_rmINTERSNP_HG19

 # Non-autosome
plink \
 --bfile KBAv2.0B_Batch*_NonAutosome_rmINTERSNP \
 --update-chr ../TOOL/updateCHRMAP.txt 2 1 \
 --update-map ../TOOL/updateCHRMAP.txt 3 1 \
 --update-alleles ../TOOL/updateALLELE.txt \
 --make-bed \
 --out KBAv2.0B_Batch*_NonAutosome_rmINTERSNP_rmINTERSNP_HG19
```

○ 옵션

· --update-chr 마커 아이디별 hg19 염색체, 위치 정보가 담긴 파일 2(염색체 컬럼) 1(마커아이디 컬럼)

· --update-map 마커 아이디별 hg19 염색체, 위치 정보가 담긴 파일 3(위치 컬럼) 1(마커아이디 컬럼)

· --update-alleles 마커 아이디별 reference allele, alternative allele 정보가 담긴 파일

○ 내용

· autosome, non-autosome 각각 통합

○ 명령어의 예

```
# Autosome
plink \
 --merge-list auto_mergelist.txt \
 --make-bed \
 --out KBAv2.0AB_Autosome

# Non-autosome
plink \
 --merge-list nonauto_mergelist.txt \
 --make-bed \
 --out KBAv2.0AB_NonAutosome
```

○ 옵션

· --merge-list: autosome, non-autosome bed, bim, fam 파일 절대경로가 포함된 파일명 (구분자: 탭)

* mergelist.txt 예

```
 # Autosome
KBAv2.0A_Batch*_Autosome_rmGENO_rmHWE_rmFST.bed
KBAv2.0A_Batch*_Autosome_rmGENO_rmHWE_rmFST.bim
KBAv2.0A_Batch*_Autosome_rmGENO_rmHWE_rmFST.fam
KBAv2.0B_Batch*_Autosome_rmGENO_rmHWE_rmFST_rmINTERSNP_HG19.bed
KBAv2.0B_Batch*_Autosome_rmGENO_rmHWE_rmFST_rmINTERSNP_HG19.bim
KBAv2.0B_Batch*_Autosome_rmGENO_rmHWE_rmFST_rmINTERSNP_HG19.fam

 # Non-autosome
KBAv2.0A_Batch*_NonAutosome.bed KBAv2.0A_Batch*_NonAutosome.bim
KBAv2.0A_Batch*_NonAutosome.fam
KBAv2.0B_Batch*_NonAutosome_rmINTERSNP_rmINTERSNP_HG19bed
KBAv2.0B_Batch*_NonAutosome_rmINTERSNP_rmINTERSNP_HG19.bim
KBAv2.0B_Batch*_NonAutosome_rmINTERSNP_rmINTERSNP_HG19.fam
```

3. 통합 후 autosome SNP 정도관리

○ 내용

· Autosome 변이에 대해서만 call rate가 낮은 SNP 및 HWE test P-value가 매우 낮은 SNP 제거

○ 기준

· Call rate가 90% 미만인 경우 제거

· HWE P-value가 10-6 미만인 경우 제거

○ 명령어의 예

```
plink \
 --bfile KBAv2.0AB_Autosome \
 --geno 0.1 \
 --hwe 1e-06 \
 --make-bed \
 --out KBAv2.0AB_Autosome_SNPQC
```

○ 옵션

· --geno: missing rate 기준 이상 제거

· --hwe: p-value 기준 미만 제거

4. 최종 통합

○ 내용

· 칩간 통합 후, autosome 변이에 대해서만 call rate가 낮은 SNP 및 HWE test P-value가 매우 낮은 SNP 제거

○ 명령어의 예

```
plink \
 --bfile KBAv2.0AB_Autosome_SNPQC \
 --bmerge KBAv2.0AB_NonAutosome \
 --make-bed \
 --out KBAv2.0AB_final
```