GenotypeFilesConvertor

Release 1.0

Muhammad Muneeb

Dec 09, 2021

CONTENTS

1 Content **3**

1.1 VCF . . . 3
1.2 BED-BIM-FAM . . . 5
1.3 PED-MAP . . . 6
1.4 GEN-SAMPLE . . . 8
1.5 23andme . . . 9
1.6 HAPS-LEGEND-SAMPLE . . . 13
1.7 RAW . . . 15
1.8 AncestryDNA . . . 18

Sample dataset used for each format is available on Google Drive.

CHAPTER

ONE

CONTENT

1.1 VCF

1.1.1 VCF to PED-MAP

```
# Include extension of file. For example, input_file.vcf.
./plink --vcf input_file.vcf --recode --out output_file
```

1.1.2 VCF to RAW

```
# Include extension of file. For example, input_file.vcf.
./plink --vcf input_file.vcf --recodeA --out output_file
```

1.1.3 VCF to BED-BIM-FAM

```
# Include extension of file. For example, input_file.vcf.
./plink --vcf input_file.vcf --make-bed --out output_file
```

1.1.4 VCF to GEN-SAMPLE

```
# Include extension of file. For example, input_file.vcf.
./plink --vcf input_file.vcf  --export oxford --out output_file
```

1.1.5 VCF to 23andme

```
# Input file should not include extension
# Make a directory in which 23andme files will be saved.
if not os.path.isdir("23andme"):
   os.mkdir("23andme")

#Convert VCF to BED_BIM_FAM --> VCFtoBED_BIM_FAM()
./plink --vcf input_file+".vcf" --make-bed --out output_file
```

(continues on next page)

(continued from previous page)

```
# It will generate three files output_file.bed, output_file.fam, and output_file.bim

#Extract id of each person
os.system("bcftools query -l "+input_file+" > ./23andme/temp_samples.txt")

#Open that file
f = open("./23andme/temp_samples.txt", "r")
for x in f:
   #Write each person name in a specific file
   temp = open("./23andme/temp.txt", "w")

   temp.write(x.strip('\n').split("_")[0] +"  "+x.strip('\n').split("_")[1])
   temp.close()

   #Extract each person from BED,BIM,FAM file and convert it to 23andme.
   os.system("./plink --bfile "+input_file.split(".")[0]+" --keep ./23andme/temp.txt --
↪recode 23 --snps-only --out ./23andme/"+x.strip('\n'))
```

1.1.6 VCF to AncestryDNA

```
#Input file should not include extension
Convert VCF to 23andme --> VCFto23andme()

#Convert 23andme to AncestryDNA -->23andmeytoAncestryDNA()
# Make a directory in which AncestryDNA files will be saved.
if not os.path.isdir("AncestryDNA"):
   os.mkdir("AncestryDNA")
  #VCFto_23andme(input_file)

#Read 23andme files
_23andmefiles  = os.listdir('./23andme')

#Read files one-by-one
for files in _23andmefiles:
   # 23andme files are in .txt file format
   if ".txt" in files and "temp" not in files:

      #Check size
      if os.stat("./23andme"+os.sep+files).st_size == 0:
         continue
      else:
         data = pd.read_csv("./23andme"+os.sep+files,sep="\t",skiprows=8)
         new = pd.DataFrame()


         new['Rsid'] = data['# rsid'].values
         new['Chromosome'] = data['chromosome'].values
         new['position'] = data['position'].values

         #Split genotype into allele1 and allele2
```

(continues on next page)

(continued from previous page)

```
        new['allele1'] = data['genotype'].str[0]
        new['allele2'] =data['genotype'].str[1]

        #Change chromosome numbers
        new['Chromosome'] = new['Chromosome'].replace(23, 'X')
        new['Chromosome'] = new['Chromosome'].replace(24, 'Y')
        new['Chromosome'] = new['Chromosome'].replace(25, 'XY')
        new['Chromosome'] = new['Chromosome'].replace(26, 'MT')
        new.to_csv("./AncestryDNA"+os.sep+files, sep="\t")
```

1.1.7 VCF to HAPS-LEGEND-SAMPLE

```
bcftools convert input_file.vcf  -h  output_file
```

1.2 BED-BIM-FAM

1.2.1 BED-BIM-FAM to PED-MAP

```
./plink --bfile input_file --recode --out output_file
```

1.2.2 BED-BIM-FAM to RAW

```
./plink --bfile input_file --recodeA --out output_file
```

1.2.3 BED-BIM-FAM to VCF

```
./plink --bfile input_file --recode vcf --out output_file
```

1.2.4 BED-BIM-FAM to GEN-SAMPLE

```
./plink --bfile input_file  --export oxford --out output_file
```

1.2.5 BED-BIM-FAM to 23andme

```
#Input file should not include extension.
if not os.path.isdir("23andme"):
   os.mkdir("23andme")

#Extract id of each person
data = pd.read_csv(input_file+".fam",sep="\s+",header=None)
print(data)
```

(continues on next page)

(continued from previous page)

```
data = data [[0,1]]
data.to_csv("./23andme/temp_samples.txt",header=False,index=False,sep=" ")

#Open that file
f = open("./23andme/temp_samples.txt", "r")
for x in f:

   #Write each person name in a specific file
   temp = open("./23andme/temp.txt", "w")
   temp.write(x)
   temp.close()

   #Extract each person from BED,BIM,FAM file and convert it to 23andme.
   os.system("./plink --bfile "+input_file.split(".")[0]+" --keep ./23andme/temp.txt --
↪recode 23 --snps-only --out ./23andme/"+x.split(" ")[0]+"_"+x.split(" ")[0])
```

1.2.6 BED-BIM-FAM to AncestryDNA

```
#1. Convert BED-BIM-FAM to VCF --> BED-BIM-FAMtoVCF()
./plink --bfile input_file --recode vcf --out output_file

#2. Convert VCF to AncestryDNA --> VCFtoAncestryDNA()
See VCFtoAncestryDNA
```

1.2.7 BED-BIM-FAM to HAPS-LEGEND-SAMPLE

```
#Input file should not include extension

#Convert BED-BIM-FAM to VCF --> BED-BIM-FAMtoVCF()
./plink --bfile input_file --recode vcf --out output_file

#Convert VCF to HAPS-LEGEND-SAMPLE --> VCFtoHAPS-LEGEND-SAMPLE()
bcftools convert output_file.VCF  -h  output_file2
```

1.3 PED-MAP

1.3.1 PED-MAP to VCF

```
./plink --file input_file --recode vcf --out output_file
```

1.3.2 PED-MAP to RAW

```
./plink --file input_file --recodeA --out output_file
```

1.3.3 PED-MAP to BED-BIM-FAM

```
./plink --file input_file --make-bed --out output_file
```

1.3.4 PED-MAP to GEN-SAMPLE

```
./plink --file  input_file --export oxford --out output_file
```

1.3.5 PED-MAP to 23andme

```
#Input file should not include extension
1. Convert PED-MAP to BED-BIM-FAM  --> PED-MAPtoBED-BIM-FAM()
./plink --file input_file --make-bed --out output_file

2. Convert BED-BIM-FAM to 23andme --> BED-BIM-FAMto23andme()
See BED-BIM-FAMto23andme
```

1.3.6 PED-MAP to AncestryDNA

```
#Input file should not include extension
1. Convert PED-MAP to VCF --> PED-MAPtoVCF()
./plink --file input_file --recode vcf --out output_file

2. Convert VCF to AncestryDNA --> VCFtoAncestryDNA()
See VCFtoAncestryDNA
```

1.3.7 PED-MAP to HAPS-LEGEND-SAMPLE

```
#Input file should not include extension
1. Convert PED-MAP to VCF --> PED-MAPtoVCF()
./plink --file input_file --recode vcf --out output_file

2. Convert VCF to HAPS-LEGEND-SAMPLE --> VCFtoHAPS-LEGEND-SAMPLE()
bcftools convert output_file.vcf  -h  output_file2
```

1.4 GEN-SAMPLE

1.4.1 GEN-SAMPLE to PED-MAP

```
./gtool  -G --g input_file.gen --s input_file.sample --ped output_file.ped --map output_
↪file.map
```

1.4.2 GEN-SAMPLE to RAW

```
./gtool  -G --g input_file.gen --s input_file.sample --ped output_file.ped --map output_
↪file.map
./plink --file output_file --recodeA --out output_file2
```

1.4.3 GEN-SAMPLE to BED-BIM-FAM

```
./gtool  -G --g input_file.gen --s input_file.sample --ped output_file.ped --map output_
↪file.map
./plink --file output_file --make-bed --out output_file2
```

1.4.4 GEN-SAMPLE to VCF

```
./gtool  -G --g input_file.gen --s input_file.sample --ped output_file.ped --map output_
↪file.map
./plink --file output_file --recode vcf --out output_file2
```

1.4.5 GEN-SAMPLE to 23andme

```
./gtool  -G --g input_file.gen --s input_file.sample --ped output_file.ped --map output_
↪file.map
./plink --file output_file --recode vcf --out output_file2
1. Convert VCF to 23andme --> VCFto23andme()
See VCFto23andme
```

1.4.6 GEN-SAMPLE to AncestryDNA

```
./gtool  -G --g input_file.gen --s input_file.sample --ped output_file.ped --map output_
↪file.map
./plink --file output_file --recode vcf --out output_file2
1. Convert VCF to 23andme --> VCFto23andme()
See VCFto23andme
2. Convert 23andme to AncestryDNA --> 23andmetoAncestryDNA()
See 23andmetoAncestryDNA
```

1.4.7 GEN-SAMPLE to HAPS-LEGEND-SAMPLE

```
./gtool  -G --g input_file.gen --s input_file.sample --ped output_file.ped --map output_
↪file.map
./plink --file output_file --recode vcf --out output_file2
bcftools convert output_file2.vcf  -h  output_file3
```

1.5 23andme

1.5.1 23andme to PED-MAP

```
#1. Convert 23andme to BED-BIM-FAM --> 23andmetoBED-BIM-FAM()
#That function will generates three files 23andmetoBED.bed, 23andmetoBED.bim, and␣
↪23andmetoBED.fam
# Step 2
./plink --bfile 23andmetoBED --recode --out output_file
```

1.5.2 23andme to RAW

```
#1. Convert 23andme to BED-BIM-FAM --> 23andmetoBED-BIM-FAM()
#That function will generates three files 23andmetoBED.bed, 23andmetoBED.bim, and␣
↪23andmetoBED.fam
# Step 2
./plink --bfile 23andmetoBED --recodeA --out output_file
```

1.5.3 23andme to BED-BIM-FAM

```
# Place 23andme files in a new directory.
# input_file is actually input directory.

if not os.path.isdir(input_file):
   print("Directory "+input_file+" does not exists...! Kindly place 23andme files in␣
↪that directory.")
   exit(0)
files = []
allfiles = os.listdir("./"+input_file+"/")
personname = []
sexinfo = "0"
for loop in allfiles:
   if ".23andme" in loop:

      data = loop.split(".")[0].split("_")
      personname.append(data[0])
      print(data)
      if data[5] =="XX":
         sexinfo = "2"
      elif data[5] =="XY":
```

(continues on next page)

(continued from previous page)

```
         sexinfo = "1"
      else:
         sexinfo = "0"
      #os.rename("tutorialsdir","tutorialsdirectory")
      os.system("./plink --23file ./"+input_file+os.sep+loop+" --snps-only --make-bed  --
↪out ./"+input_file+os.sep+data[0])
      if os.path.exists("./"+input_file+os.sep+data[0]+".fam"):
         data2 = pd.read_csv("./"+input_file+os.sep+data[0]+".fam",header=None, sep="\s+
↪")
         data2[0] = data[0]
         data2[1] = data[0]
         data2[4] = sexinfo
         data2.to_csv("./"+input_file+os.sep+data[0]+".fam",sep="\t",header=False,
↪index=False)
allfiles =  os.listdir("./"+input_file+"/")
count=0
files=[]
for loop in allfiles:
   if ".txt" in loop and ".bed" not in loop and ".fam" not in loop and ".bim" not in␣
↪loop:
      print(loop)
      x = loop.split("_")[0]
      x = x + ".fam"
      me = os.path.exists("./"+input_file+os.sep+x)
      if me==True:
         x = x.split(".")[0]
         x = "./"+input_file+os.sep+x +".bed " + "./"+input_file+os.sep+x + ".bim " +  ".
↪/"+input_file+os.sep+x + ".fam"
         files.append(x)
      else:
         count=count+1
      print(count," People removed due to missing fam file")
with open("./"+input_file+os.sep+"All.txt", "w") as filehandle:
   for listitem in files:
     filehandle.write('%s\n' % listitem)

os.system("./plink --merge-list ./"+input_file+os.sep+"/All.txt --make-bed --out␣
↪23andmetoBED")

if os.path.exists("23andmetoBED.bed"):
   exit(0)
else:
   allfiles =  os.listdir("./"+input_file+"/")
   count=0
   for loop in allfiles:
      if ".bed" in loop:
         x = loop
         x = x.split(".")[0]

         command = "./plink --bfile ./"+input_file+os.sep+x+" --exclude 23andmetoBED-
↪merge.missnp --make-bed --out ./"+input_file+os.sep + x
         os.system(command)
```

(continues on next page)

(continued from previous page)

```
   allfiles =  os.listdir("./"+input_file+"/")
   files=[]

   for loop in allfiles:
      if ".txt" in loop and ".bed" not in loop and ".fam" not in loop and ".bim" not in␣
↪loop:
         print(loop)
         x = loop.split("_")[0]
         x = x + ".fam"
         me = os.path.exists("./"+input_file+os.sep+x)
         if me==True:
            x = x.split(".")[0]
            x = "./"+input_file+os.sep+x +".bed " + "./"+input_file+os.sep+x + ".bim " +␣
↪ "./"+input_file+os.sep+x + ".fam"
            files.append(x)
         else:
            count=count+1
         print(count," People removed due to missing fam file")
   with open("./"+input_file+os.sep+"All.txt", "w") as filehandle:
     for listitem in files:
         filehandle.write('%s\n' % listitem)
   os.system("./plink --merge-list ./"+input_file+os.sep+"/All.txt --make-bed --out␣
↪23andmetoBED")
```

1.5.4 23andme to GEN-SAMPLE

```
#1. Convert 23andme to BED-BIM-FAM --> 23andmetoBED-BIM-FAM()
#That function will generates three files 23andmetoBED.bed, 23andmetoBED.bim, and␣
↪23andmetoBED.fam
# Step 2
./plink --bfile 23andmetoBED  --export oxford --out output_file
```

1.5.5 23andme to VCF

```
#1. Convert 23andme to BED-BIM-FAM --> 23andmetoBED-BIM-FAM()
#That function will generates three files 23andmetoBED.bed, 23andmetoBED.bim, and␣
↪23andmetoBED.fam
# Step 2
./plink --bfile 23andmetoBED --recode vcf --out output_file
```

1.5.6 23andme to AncestryDNA

```
# Make a directory in which AncestryDNA files will be saved.
if not os.path.isdir("AncestryDNA"):
   os.mkdir("AncestryDNA")
_23andmefiles  = os.listdir("./"+input_file)


#Read files one-by-one
for files in _23andmefiles:
   # 23andme files are in .txt file format
   if "23andme.txt" in files and "temp" not in files:

      #Check size
      if os.stat("./"+input_file+os.sep+files).st_size == 0:
         continue
      else:
         print(files)
         data = pd.read_csv("./"+input_file+os.sep+files,sep="\t", comment='#',
↪header=None,low_memory=False)
         new = pd.DataFrame()
         new['Rsid'] = data[0].values
         new['Chromosome'] = data[1].values
         new['position'] = data[2].values


         #Split genotype into allele1 and allele2
         new['allele1'] = data[3].str[0]
         new['allele2'] =data[3].str[1]

         #Change chromosome numbers
         new['Chromosome'] = new['Chromosome'].replace(23, 'X')
         new['Chromosome'] = new['Chromosome'].replace(24, 'Y')
         new['Chromosome'] = new['Chromosome'].replace(25, 'XY')
         new['Chromosome'] = new['Chromosome'].replace(26, 'MT')


         #Rename file name
         files = files.replace("23andme","ancestry")
         new.to_csv("./AncestryDNA"+os.sep+files, sep="\t",index=False)
```

1.5.7 23andme to HAPS-LEGEND-SAMPLE

```
#1. Convert 23andme to BED-BIM-FAM --> 23andmetoBED-BIM-FAM()
#That function will generates three files 23andmetoBED.bed, 23andmetoBED.bim, and␣
↪23andmetoBED.fam
# Step 2
./plink --bfile 23andmetoBED --recode vcf --out output_file

# Step 3
bcftools convert output_file.vcf  -h  output_file2
```

1.6 HAPS-LEGEND-SAMPLE

We used files from hapmap3_r2_b36. to demonstrate the conversion to other formats.

1. There is a separate file for each chromosome, and we have to modify the X.legend file before further processing.
2. X.Haps and X.legend file should be compressed in gz format before further processing.

LEGEND file before

```
rsID position a0 a1
rs10458597 554484 C T
rs2185539 556738 C T
rs11240767 718814 C T
rs12564807 724325 A G
rs3131972 742584 A G
rs3131969 744045 A G
rs3131967 744197 C T
rs1048488 750775 C T
rs12562034 758311 A G
```

LEGEND file after

```
ID      position        a0      a1
1:554484_C_T    554484  C       T
1:556738_C_T    556738  C       T
1:718814_C_T    718814  C       T
1:724325_A_G    724325  A       G
1:742584_A_G    742584  A       G
1:744045_A_G    744045  A       G
1:744197_C_T    744197  C       T
1:750775_C_T    750775  C       T
1:758311_A_G    758311  A       G
```

1.6.1 HAPS-LEGEND-SAMPLE to PED-MAP

```
# Step 1.  Convert HAPS-LEGEND-SAMPLE to VCF.
# It will generate one file "output_file.vcf"
# Step 2.
./plink --vcf output_file --recode --out output_file2
```

1.6.2 HAPS-LEGEND-SAMPLE to RAW

```
# Step 1.  Convert HAPS-LEGEND-SAMPLE to VCF.
# It will generate one file "output_file.vcf"
# Step 2.
./plink --vcf output_file --recodeA --out output_file2
```

1.6.3 HAPS-LEGEND-SAMPLE to BED-BIM-FAM

```
# Step 1.  Convert HAPS-LEGEND-SAMPLE to VCF.
# It will generate one file "output_file.vcf"
# Step 2.
./plink --vcf output_file --make-bed --out output_file2
```

1.6.4 HAPS-LEGEND-SAMPLE to GEN-SAMPLE

```
# Step 1.  Convert HAPS-LEGEND-SAMPLE to VCF.
# It will generate one file "output_file.vcf"
# Step 2.
./plink --vcf output_file --export oxford --out output_file2
```

1.6.5 HAPS-LEGEND-SAMPLE to 23andme

```
# Step 1.  Convert HAPS-LEGEND-SAMPLE to VCF.
# It will generate one file "output_file.vcf"
if not os.path.isdir("23andme"):
   os.mkdir("23andme")

VCFtoBED_BIM_FAM("output_file.vcf")
os.system("bcftools query -l output_file.vcf > ./23andme/temp_samples.txt")
f = open("./23andme/temp_samples.txt", "r")
for x in f:
   temp = open("./23andme/temp.txt", "w")
   print("X")
   temp.write(x.strip('\n')+" "+x.strip('\n'))
   temp.write('\n')

   temp.close()
    os.system("./plink --bfile output_file --keep ./23andme/temp.txt --recode 23 --snps-
↪only --out ./23andme/"+x.strip('\n')+".23andme")
```

1.6.6 HAPS-LEGEND-SAMPLE to AncestryDNA

```
# Step 1.  HAPS-LEGEND-SAMPLE to 23andme
# Step 2. 23andme to AncestryDNA
See (23andme to AncestryDNA)
```

1.6.7 HAPS-LEGEND-SAMPLE to VCF

```
data = pd.read_csv("hapmap3_r2_b36_chr1.legend",index_col=False, sep="\s+")
#Create a new ID column for legend file
data["ID"] = "1:"+data['position'].astype(str)+"_"+data["a0"]+"_"+data["a1"]
data =data[['ID','position','a0','a1']]
data.to_csv("hapmap3_r2_b36_chr1.legend",index=False, sep="\t")

#Zip the legend file
os.system("gzip hapmap3_r2_b36_chr1.legend")
#Rename hapmap3_r2_b36_chr1.haps to hapmap3_r2_b36_chr1.hap
os.rename("hapmap3_r2_b36_chr1.haps.gz","hapmap3_r2_b36_chr1.hap.gz")

#Rename hapmap3_r2_b36_all.sample to hapmap3_r2_b36_chr1.samples
os.rename("hapmap3_r2_b36_all.sample","hapmap3_r2_b36_chr1.samples")

os.system("bcftools convert  --haplegendsample2vcf  hapmap3_r2_b36_chr1  -o  output_file.
↪vcf")
```

1.7 RAW

1.7.1 RAW to PED-MAP

```
def converter(x):
   # Fill "NA" with '0 0'
   x = x.fillna('0 0')

   # Convert numbers to integer
   x.astype(int, errors='ignore')
   ref = x.name[-1]

   # Encoding of PED file
   if ref=="G":
     x = x.replace(0, "G G")
     x = x.replace(1, "G C")
     x = x.replace(2, "C C")

   if ref=="C":
     x = x.replace(0, "C C")
     x = x.replace(1, "C G")
     x = x.replace(2, "G G")

   if ref=="T":
     x = x.replace(0, "T T")
     x = x.replace(1, "T A")
     x = x.replace(2, "A A")

   if ref=="A":
     x = x.replace(0, "A A")
     x = x.replace(1, "A T")
```

(continues on next page)

(continued from previous page)

```
    x = x.replace(2, "T T")


  return x


# Extract SNPs names, which is in this format SNP_REFAllele
#os.system("cat "+input_file+" | head -n 1  >> snps.txt")
print("cat "+input_file+" | head -n 1  >> snps.txt")


data = pd.read_csv("snps.txt",index_col=None,header=None,sep="\s+").loc[:, 6:].T

# Make a directory to store chunks
# Chunking is required because RAW file is usually large in size
if not os.path.isdir("Chunks"):
  os.mkdir("Chunks")

# Make ".MAP" file
# RAW file does not contain the position and chromosome number information so, all other␣
↪columns except 2nd are 0.
maps = pd.DataFrame()
maps[0] = [0]*len(data)
maps[1] = data[0].values
maps[2] = [0]*len(data)
maps[3] = [0]*len(data)
maps.to_csv("final.map",sep="\t",header=False,index=False)

_smallraw  = os.listdir('./Chunks')
count=0
_smallraw = sorted(_smallraw)

# Encode each chunk which is same as that of ped file.
for files in _smallraw:
  if ".txt" not in files:
    if count==0:
      count=1
      data2 = pd.read_csv("Chunks"+os.sep+files,sep="\s+")
      data2[list(data[0].values)] = data2[list(data[0].values)].apply(converter)
      data2.to_csv("Chunks"+os.sep+files+".txt",sep="\t",index=False,header=False)

    else:
      data2 = pd.read_csv("Chunks"+os.sep+files,sep="\s+",names=list(data2.columns.
↪values))
      data2[list(data[0].values)] = data2[list(data[0].values)].apply(converter)
      data2.to_csv("Chunks"+os.sep+files+".txt",sep="\t",index=False,header=False)
final = pd.DataFrame()
#Merge all chunks
for files in _smallraw:
  if ".txt" in files:
    if count==0:
      count=1
```

(continues on next page)

(continued from previous page)

```
        final = pd.read_csv("Chunks"+os.sep+files,sep="\t",index_col=None,low_
↪memory=False,header=None)
    else:
        data2 = pd.read_csv("Chunks"+os.sep+files,sep="\t",header=None,index_col=None,
↪low_memory=False)
        final = final.append(data2, ignore_index=True)
        del data2
final.to_csv("final.ped",sep="\t",index=False,header=None)

# After this step you will have two files: final.ped and final.map
```

1.7.2 RAW to VCF

```
#Step 1. Convert Raw file to PED-MAP. See RAWtoPED-MAP.
It generates two files: final.ped and final.map
#Step 2.
./plink --file final --recode vcf --out output_file
```

1.7.3 RAW to BED-BIM-FAM

```
#Step 1. Convert Raw file to PED-MAP. See RAWtoPED-MAP.
It generates two files: final.ped and final.map
#Step 2.
./plink --file final  --make-bed --out output_file
```

1.7.4 RAW to GEN-SAMPLE

```
#Step 1. Convert Raw file to PED-MAP. See RAWtoPED-MAP.
It generates two files: final.ped and final.map
#Step 2.
./plink --file final  --export oxford --out output_file
```

1.7.5 RAW to 23andme

```
#Step 1. Convert Raw file to PED-MAP. See RAWtoPED-MAP.
It generates two files: final.ped and final.map

#Step 2. Convert PED-MAP to BED-BIM-FAM  --> PED-MAPtoBED-BIM-FAM()
./plink --file input_file --make-bed --out output_file

2. Convert BED-BIM-FAM to 23andme --> BED-BIM-FAMto23andme()
See BED-BIM-FAMto23andme
```

1.7.6 RAW to AncestryDNA

```
#Step 1. Convert Raw file to PED-MAP. See RAWtoPED-MAP.
It generates two files: final.ped and final.map

#Step 2. Convert PED-MAP to BED-BIM-FAM  --> PED-MAPtoBED-BIM-FAM()
./plink --file input_file --make-bed --out output_file

#Step 3. Convert BED-BIM-FAM to 23andme --> BED-BIM-FAMto23andme()
See BED-BIM-FAMto23andme

#Step 4. Convert 23andme to AncestryDNA --> 23andmetoAncestryDNA()
See 23andmetoAncestryDNA
```

1.7.7 RAW to HAPS-LEGEND-SAMPLE

```
#Step 1. Convert Raw file to PED-MAP.
It generates two files: final.ped and final.map
#Step 2. Convert PED-MAP file to VCF.
./plink --file  final --recode vcf --out output_file
#Step 3. Convert VCF file to HAPS-LEGEND-SAMPLE.
bcftools convert output_file.vcf  -h  output_file2
```

1.8 AncestryDNA

1.8.1 AncestryDNA to PED-MAP

```
#1. Convert AncestryDNA files to 23andme --> AncestryDNAto23andme()
#2. Convert 23andme files to BED-BIM-FAM --> 23andmetoBED-BIM-FAM()
#That function will generates three files 23andmetoBED.bed, 23andmetoBED.bim, and␣
↪23andmetoBED.fam
# Step 3
./plink --bfile 23andmetoBED --recode vcf --out output_file
```

1.8.2 AncestryDNA to RAW

```
#1. Convert AncestryDNA files to 23andme --> AncestryDNAto23andme()
#2. Convert 23andme files to BED-BIM-FAM --> 23andmetoBED-BIM-FAM()
#That function will generates three files 23andmetoBED.bed, 23andmetoBED.bim, and␣
↪23andmetoBED.fam
# Step 3
./plink --bfile 23andmetoBED --recodeA --out output_file
```

1.8.3 AncestryDNA to BED-BIM-FAM

```
#1. Convert AncestryDNA files to 23andme --> AncestryDNAto23andme()
#2. Convert 23andme files to BED-BIM-FAM --> 23andmetoBED-BIM-FAM()
#That function will generates three files 23andmetoBED.bed, 23andmetoBED.bim, and␣
↪23andmetoBED.fam
```

1.8.4 AncestryDNA to GEN-SAMPLE

```
#1. Convert AncestryDNA files to 23andme --> AncestryDNAto23andme()
#2. Convert 23andme files to BED-BIM-FAM --> 23andmetoBED-BIM-FAM()
#That function will generates three files 23andmetoBED.bed, 23andmetoBED.bim, and␣
↪23andmetoBED.fam
# Step 3
./plink --bfile 23andmetoBED  --export oxford --out output_file
```

1.8.5 AncestryDNA to 23andme

```
#Make a directory in which 23andme files will be saved.
if not os.path.isdir("23andme"):
   os.mkdir("23andme")

#Read AncestryDNA files from the directory
_ancestry  = os.listdir('./AncestryDNA')
for files in _ancestry:

   if ".txt" in files and "temp" not in files:
      if os.stat("./AncestryDNA"+os.sep+files).st_size == 0:
      continue
   else:
      data = pd.read_csv("./AncestryDNA"+os.sep+files,sep="\t",skiprows=18)

      new = pd.DataFrame()
      new['Rsid'] = data['rsid'].values
      new['Chromosome'] = data['chromosome'].values
      new['position'] = data['position'].values

      #Merge genotype data
      new['genotype'] = data['allele2']+ data['allele1']
      new['Chromosome'] = new['Chromosome'].replace(23, 'X')
      new['Chromosome'] = new['Chromosome'].replace(24, 'Y')
      new['Chromosome'] = new['Chromosome'].replace(25, 'XY')
      new['Chromosome'] = new['Chromosome'].replace(26, 'MT')

      files = files.replace("ancestry","23andme")

      #Save each file in "23andme" directory
      new.to_csv("./23andme"+os.sep+files, sep="\t",index=False,header=False)
```

1.8.6 AncestryDNA to VCF

```
#1. Convert AncestryDNA files to 23andme --> AncestryDNAto23andme()
#2. Convert 23andme files to BED-BIM-FAM --> 23andmetoBED-BIM-FAM()
#That function will generates three files 23andmetoBED.bed, 23andmetoBED.bim, and␣
↪23andmetoBED.fam
# Step 3
./plink --bfile 23andmetoBED --recode vcf --out output_file
```

1.8.7 AncestryDNA to HAPS-LEGEND-SAMPLE

```
#1. Convert AncestryDNA files to 23andme --> AncestryDNAto23andme()
#2. Convert 23andme files to BED-BIM-FAM --> 23andmetoBED-BIM-FAM()
#That function will generates three files 23andmetoBED.bed, 23andmetoBED.bim, and␣
↪23andmetoBED.fam
# Step 3
./plink --bfile 23andmetoBED --recode vcf --out output_file

# Step 4
bcftools convert output_file.vcf  -h  output_file2
```