Manually editing PLINK files
R version

Daniel Howrigan

Data Group Meeting

January 12, 2021

Tutorial questions

Problem #1: Manipulating PLINK .fam file

Background: Converting from VCF to PLINK creates a stripped-down .fam file where only the individual ID (IID) column is populated. We want to use data in the phenotype file (often provided by the project manager) to fill in the .fam file

Problem #2: Manipulating PLINK .bim file

Background: Want to merge data with thousand genomes (1KG) reference dataset to run PCA analysis. Custom SNP IDs are causing errors, so we want to see what SNPs overlap using the genomic position/alleles. Among overlapping SNPs, replace custom SNP IDs with the reference SNP ID (rsID) used in 1KG.

Files available here: https://personal.broadinstitute.org/howrigan/tutorial/

R script:

```
## creating hypothetical chipwell identifiers from a genotyping
plate
## using seq() to create a sequence, sprintf() to have leading
zeros, and paste0() to combine into a single character
rows <- paste0('R',sprintf("%02d",seq(1,12,1)))
cols <- paste0('C',sprintf("%02d",seq(1,8,1)))

## create empty individual ID variable
IID <- c()
## create counter to accumulate through multiple for() loops
counter <- 1

## loop through rows object
for (a in 1:length(rows)) {
    ## loop through column object
    for (b in 1:length(cols)) {
      ##  combine row and column character
      IID[counter] <- paste0(rows[a],cols[b])
      ## add to counter
      counter <- counter + 1
  } # b loop
} # a loop

## add an additional barcode identifier to the "chipwell"
IID <- paste0(IID,'_1000',seq(1,length(IID),1))


## Now create a .fam file that PLINK would create from a VCF
## create zero vector the length of IID
zeros <- rep(0,length(IID))
## create -9 vector (PLINK recognizes -9 as missing) the length of
IID
negnine <- rep(-9,length(IID))

## combine vectors into a data frame resembling a PLINK .fam file
fam <- cbind.data.frame(zeros,IID,zeros,zeros,zeros,negnine)
## create default column names when R reads in a file with no header
(V1, V2, etc..)
names(fam) <- paste0('V',seq(1,6,1))
```

Creating an example .fam file

Creating a typical .fam file from VCF using base R commands.

Functions used:

- paste() = combine strings & objects
- paste0() = omit space delimiter
- sprintf() = print character string
- for() = iterate through variable
- seq() = sequence of numbers
- c() = concatenate
- rep() = repeat strings
- length() = get length of vector
- cbind.data.frame() = combine vectors into a data frame

```
> head(fam)
  V1            V2 V3 V4 V5 V6
1  0 R01C01_10001  0  0  0 -9
2  0 R01C02_10002  0  0  0 -9
3  0 R01C03_10003  0  0  0 -9
4  0 R01C04_10004  0  0  0 -9
5  0 R01C05_10005  0  0  0 -9
6  0 R01C06_10006  0  0  0 -9
```

R script:

```
## create phenotype file with linked IDs

chipwell_barcode <- IID
PT_ID <- paste0('PT-000',seq(1,length(IID),1))
SM_ID <- paste0('SM-000',seq(1,length(IID),1))
collaborator_sample_ID <-
paste0('sample',seq(1,length(IID),1))

indx <- rbinom(length(IID),1,0.5)
disease <- rep('case',length(IID))
disease[indx==0] <- 'control'

indx <- rbinom(length(IID),1,0.5)
sex <- rep('male',length(IID))
sex[indx==0] <- 'female'

phe <-
cbind.data.frame(PT_ID,SM_ID,collaborator_sample_ID,chipwell_
barcode,disease,sex)



## write fam and phe data frames to file
write.table(fam,"test.fam",col=F,row=F,quo=F,sep='\t')
write.table(phe,"test.phe",col=T,row=F,quo=F,sep='\t')

## make a copy of original fam file using system() to run on
command line
system('cp test.fam test.vcf_conversion.fam')
```

Creating an example .phe file

Creating a typical phenotype file received by Broad project management.

Functions used:

- rbinom() = binomial sampling
- write.table() = write object to file
- system() = run on command line

```
> head(phe[,1:3])
    PT_ID   SM_ID collaborator_sample_ID
1 PT-0001 SM-0001                sample1
2 PT-0002 SM-0002                sample2
3 PT-0003 SM-0003                sample3
4 PT-0004 SM-0004                sample4
5 PT-0005 SM-0005                sample5
6 PT-0006 SM-0006                sample6

> head(phe[,4:6])
  chipwell_barcode disease    sex
1     R01C01_10001    case female
2     R01C02_10002 control female
3     R01C03_10003 control female
4     R01C04_10004 control female
5     R01C05_10005 control   male
6     R01C06_10006    case   male
```

R script:

```
## read in fam and phe data frames
fam <- read.table('test.fam',stringsAsFactors = F)
phe <- read.table('test.phe',h=T,stringsAsFactors = F)

## Update fam file

## IMPORTANT: create order column to preserve order of .fam file
fam$order <- seq(1,nrow(fam),1)

## merge by chipwell_barcode column (V2 in .fam file,
chipwell_barcode in .phe file)
mrg <- merge(fam,phe,by.x='V2',by.y='chipwell_barcode',all=T)

## here is a good spot to check if any IDs are present only in one
file and not the other

## update sex (1=male; 2=female; other=unknown)
mrg$sex2 <- 0
mrg$sex2[mrg$sex=='male'] <- 1
mrg$sex2[mrg$sex=='female'] <- 2

## update affection status (0=missing,1=unaffected, 2=affected, -
9=missing)
mrg$aff <- 0
mrg$aff[mrg$disease=='control'] <- 1
mrg$aff[mrg$disease=='case'] <- 2

## re-order with order() to maintain original .fam file order
mrg <- mrg[order(mrg$order),]

## select columns to create updated .fam file
fam2 <-
mrg[,c('collaborator_sample_ID','collaborator_sample_ID','V3','V4','
sex2','aff')]

## write updated .fam file
write.table(fam2,"test.fam",col=F,row=F,quo=F,sep='\t')
```

Merging .fam and .phe file to create Updated .fam file

Adding in relevant phenotype info in PLINK format

SEX = Sex (1=male; 2=female; other=unknown)
AFF = Case/control phenotype
(0=missing,1=unaffected, 2=affected, -9=missing)

Functions used:

- read.table() = read in file
- merge() = merge data frames
- object[rows,columns]
  - [index,] = keep selected rows
  - [,index] = keep selected columns

```
> head(fam2)
sample1 sample1 0 0 2 2
sample2 sample2 0 0 2 1
sample3 sample3 0 0 2 1
sample4 sample4 0 0 2 1
sample5 sample5 0 0 1 1
sample6 sample6 0 0 1 2
```

R script:

```
## read in fam and phe data frames
fam <- read.table('test.fam',stringsAsFactors = F)
phe <- read.table('test.phe',h=T,stringsAsFactors = F)

## Update fam file

## IMPORTANT: create order column to preserve order of .fam file
fam$order <- seq(1,nrow(fam),1)

## merge by chipwell_barcode column (V2 in .fam file,
chipwell_barcode in .phe file)
mrg <- merge(fam,phe,by.x='V2',by.y='chipwell_barcode',all=T)

## here is a good spot to check if any IDs are present only in one
file and not the other

## update sex (1=male; 2=female; other=unknown)
mrg$sex2 <- 0
mrg$sex2[mrg$sex=='male'] <- 1
mrg$sex2[mrg$sex=='female'] <- 2

## update affection status (0=missing,1=unaffected, 2=affected, -
9=missing)
mrg$aff <- 0
mrg$aff[mrg$disease=='control'] <- 1
mrg$aff[mrg$disease=='case'] <- 2

## re-order with order() to maintain original .fam file order
mrg <- mrg[order(mrg$order),]

## select columns to create updated .fam file
fam2 <-
mrg[,c('collaborator_sample_ID','collaborator_sample_ID','V3','V4','
sex2','aff')]

## write updated .fam file
write.table(fam2,"test.fam",col=F,row=F,quo=F,sep='\t')
```

Merging .fam and .phe file to create Updated .fam file

Adding in relevant phenotype info in PLINK format

SEX = Sex (1=male; 2=female; other=unknown)
AFF = Case/control phenotype
(0=missing,1=unaffected, 2=affected, -9=missing)

Functions used:

- read.table() = read in file
- merge() = merge data frames

```
> head(fam2)
sample1 sample1 0 0 2 2
sample2 sample2 0 0 2 1
sample3 sample3 0 0 2 1
sample4 sample4 0 0 2 1
sample5 sample5 0 0 1 1
sample6 sample6 0 0 1 2
```

Tutorial questions

Problem #1: Manipulating PLINK .fam file

Background: Converting from VCF to PLINK creates a stripped-down .fam file where only the individual ID (IID) column is populated. We want to use data in the phenotype file (often provided by the project manager) to fill in the .fam file

Problem #2: Manipulating PLINK .bim file

Background: Want to merge data with thousand genomes (1KG) reference dataset to run PCA analysis. Custom SNP IDs are causing errors, so we want to see what SNPs overlap using the genomic position/alleles. Among overlapping SNPs, replace custom SNP IDs with the reference SNP ID (rsID) used in 1KG.

Files available here: https://personal.broadinstitute.org/howrigan/tutorial/

R script:

```
## 2 - Manually updating .bim file

## GOAL:
## - find shared SNPs based on position
## - change SNP IDs to successfully merge data
## - write out shared SNP IDs

## On Broad:
setwd('/web/personal/howrigan/tutorial')

## Load data.table package to read in larger files faster
## use install.packages("data.table") to install
library(data.table)

## Read in .map files to compare
kg <- fread('pop_4pop_mix_SEQ_hg19_first50k.bim')
ff <- fread('affy_exomechip_hg19_first5k.bim')

## create positional IDs (without SNP ID)
kga <- paste0(kg$V1,'_',kg$V4,'_',kg$V5,'_',kg$V6)

## Use both A1/A2 and A2/A1 IDs
ffa <- paste0(ff$V1,'_',ff$V4,'_',ff$V5,'_',ff$V6)
ffb <- paste0(ff$V1,'_',ff$V4,'_',ff$V6,'_',ff$V5)
## combine both IDs
ffc <- c(ffa,ffb)

## See how many positional matches we can find
sum(kga %in% ffa) ## 873
sum(kga %in% ffb) ## 49
sum(kga %in% ffc) ## 922
```

Reading in .bim files and creating Genomic position IDs

Functions used:

- fread() = fast read in file
- %in% - is this vector in that vector?

```
> head(kg)
   V1          V2 V3    V4 V5 V6
1:  1 rs140337953  0 30923  G  T
2:  1  rs10399749  0 55299  T  C
3:  1  rs76735897  0 61987  G  A
4:  1 rs116440577  0 63671  A  G
5:  1  rs75062661  0 69511  A  G
6:  1 rs143773730  0 73841  T  C
```

```
> head(ff)
   V1          V2    V3     V4 V5 V6
1:  1 AX-83399130 0.001  69496  0  G
2:  1 AX-82952048 0.001  69590  0  T
3:  1 AX-83277131 0.008 762485  A  C
4:  1 AX-82936492 0.010 865628  A  G
5:  1 AX-83098582 0.010 871229  0  G
6:  1 AX-83090629 0.010 874809  C  G
```

R script:

```
## add positional IDs to .bim files (faster than appending to data frame)
kg$kg_pos <- kga
ff$ff_pos1 <- ffa
ff$ff_pos2 <- ffb

## match positional IDs of 1KG to affy and provide rsID
indx1 <- kg$V2[match(ff$ff_pos1,kg$kg_pos)]
indx2 <- kg$V2[match(ff$ff_pos2,kg$kg_pos)]
## start all as '.'
ff$rsID <- '.'
## insert only non-NA values from indx to affy rsID
ff$rsID[!is.na(indx1)] <- indx1[!is.na(indx1)]
ff$rsID[!is.na(indx2)] <- indx2[!is.na(indx2)]

## create new .bim file
ff2 <- ff[,c('V1','rsID','V3','V4','V5','V6')]

head(ff2)

## create SNP Map Key
overlap_key <- subset(ff[,c('V2','rsID')],ff$rsID != '.')
head(overlap_key)

## Options to subset/update-id in PLINK or overwrite .bim file
```

Reading in .bim files and creating Genomic position IDs

Functions used:

- match() = which values match
- !is.na() = evaluate non-NA values only
- subset() = keep values that meet selection criteria

```
> head(ff2)
 V1         rsID    V3     V4 V5 V6
 1:  1            . 0.001  69496  0  G
 2:  1            . 0.001  69590  0  T
 3:  1 rs148989274 0.008 762485  A  C
 4:  1            . 0.010 865628  A  G
 5:  1            . 0.010 871229  0  G
 6:  1            . 0.010 874809  C  G

> head(overlap_key)
            V2         rsID
1: AX-83277131 rs148989274
2: AX-32571027   rs2272757
3: AX-40388697  rs13303010
4: AX-40437185      rs1921
5: AX-83223080   rs3934834
6: AX-83234979   rs6668667
```