Script Name: GENESIS_setup_ANALYSIS_standalone.R

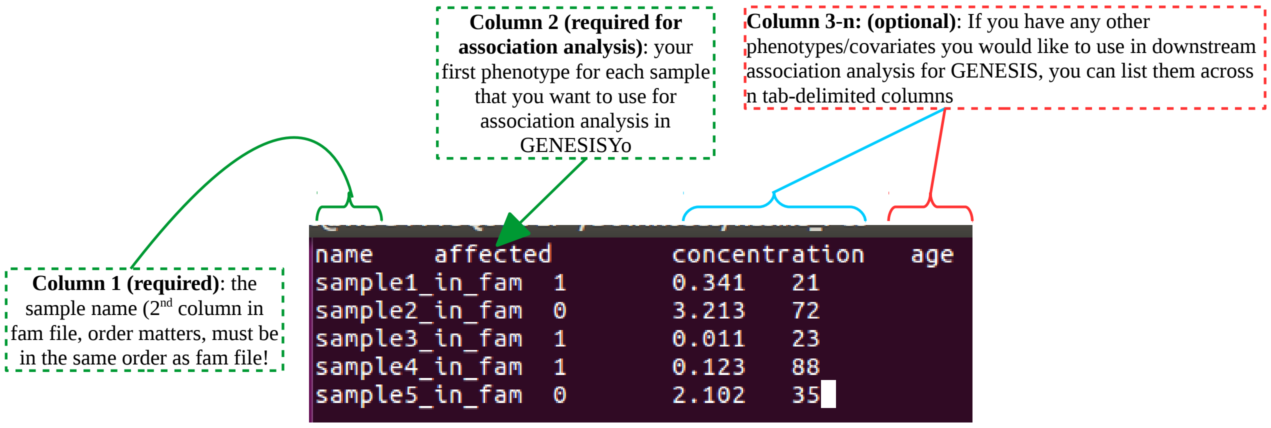
Description: Expected output is a binary Rdata object containing covariate genetic model matrix, scanAnnot data frame carrying the following: 50 PCs, any other covariate information to include, and phenotype(s), and finally the data object also contains the pcair full set of 200 PCs (if available)

General Use Instructions

- for use in interactive Rstudio Session or after all variables have been manually set and that variables have been added in step 7 then can run script on command line

Step 1:

- Generate a clean LD-pruned set of SNPs in PLINK binary format (.bed, .bim, .fam)
- Generate a tab-delimited phenotype file. This needs to be in the same sample order as your fam file (see dummy_phenoFile.txt). If other covariates should be added, they should be added here in as many columns as needed. You can name the headers anything you want as long as they exist and they contain **no whitespaces or special characters**!!



Column 2 (required for association analysis): your first phenotype for each sample that you want to use for association analysis in GENESISYo

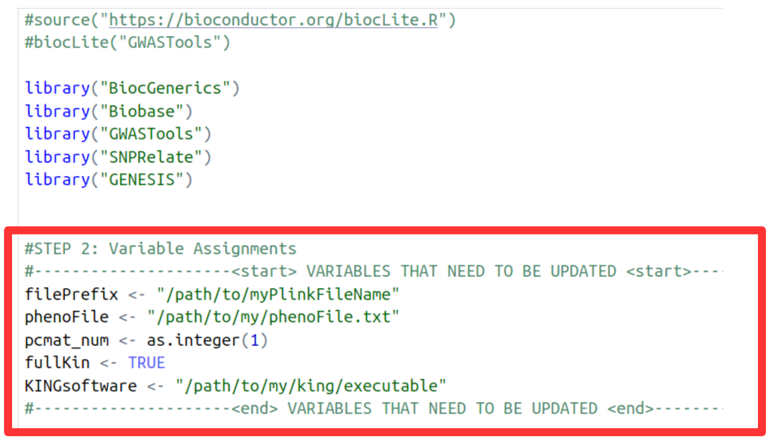
Column 3-n: (optional): If you have any other phenotypes/covariates you would like to use in downstream association analysis for GENESIS, you can list them across n tab-delimited columns

Column 1 (required): the sample name (2nd column in fam file, order matters, must be in the same order as fam file!

```
name     affected        concentration   age
sample1_in_fam  1        0.341   21
sample2_in_fam  0        3.213   72
sample3_in_fam  1        0.011   23
sample4_in_fam  1        0.123   88
sample5_in_fam  0        2.102   35
```

Step 2: Variable assignments

- Open script in Rstudio and fill in variable section at the top called: "VARIABLES THAT NEED TO BE UPDATED" Change the items in quotes for each variable. For *pcmat_num* be sure to change the default of *1* to a different integer that makes sense for your project.



```
#source("https://bioconductor.org/biocLite.R")
#biocLite("GWASTools")

library("BiocGenerics")
library("Biobase")
library("GWASTools")
library("SNPRelate")
library("GENESIS")

#STEP 2: Variable Assignments
#--------------------<start> VARIABLES THAT NEED TO BE UPDATED <start>---
filePrefix <- "/path/to/myPlinkFileName"
phenoFile <- "/path/to/my/phenoFile.txt"
pcmat_num <- as.integer(1)
fullKin <- TRUE
KINGsoftware <- "/path/to/my/king/executable"
#--------------------<end> VARIABLES THAT NEED TO BE UPDATED <end>-------
```

Variable Name	Type	Definition
filePrefix	string	The name (and path location if not in current working directory) of your plink file without the . fileExtension. When the script is run, it will look for this file in your working directory
phenoFile	string of filename	This file is tab-delimited where the first column is the sample name (2nd column of plink fam file) and any subsequent columns are phenotypes of interest; phenoFile should be in the same sample order as the input PLINK file
pcmat_num	integer	This is an integer that tells the pcmat function the total number of principal components (generated from pcair) to use for ancestry adjustment
fullKin	boolean	When set to "TRUE", PCs calculations use both the .kin [within family kinship matrix] and .kin0 [between family kinship matrix] files generated from the KING software; otherwise this should be set to FALSE and only the .kin0 file is used in PC generation for familial relationships; **NOTE: setting this to TRUE only works if FIDs are actually set to identify family groups within PLINK file**
KINGsoftware	string	Full path to installed KING executable file on your local machine

Step 3: Generating KING output

- No changes to the code need to be made here.

  Explanation of code:

  If **fullKin = TRUE**, it will run KING using the full path you specified to your KING executable (variable KINGsoftware) with an extra --*kinship* flag on the command line which will calculate and generate two files:
  - within family kinship matrix (.kin) and
  - between family kinship matrix (.kin0)

    Warning: If using fullKin = TRUE, be sure your FID values in your PLINK files are assigned to true families; KING calculates the within family kinship matrix based on FID families. Therefore, **if all FIDs are unique, this is the same as running fullKin=FALSE** because KING will not detect any within family relationships

  If **fullKin = FALSE** it will still generate two files, but the .kin file will be empty:
  - within family kinship matrix (.kin) – **EMPTY, just header**
  - between family kinship matrix (.kin0)

Step 4: File Format Conversions and Sample ID Generation

- No changes to the code need to be made here.

Explanation of code:

Converts binary PLINK files into gds format. Also for code readability, both .kin0 and .kin output from KING software are stored as variables: file.kin0 and file.kin, respectively.

The gds file is then read into a gds reader so genotypes and samples names can be extracted efficiently.

During this stage a new tab-delimited file is generated which contains the GENESIS ID number from 1 to n of all the samples in your PLINK files and matches it to the sample name in your PLINK file. This is file has the same name as your PLINK prefix input but ends with the following: "_GENESIS_ID_key_file.txt"

Step 5: Calculate PCs

- No changes to the code need to be made here.

Explanation of code:

Checks if using both within-family population structure and between family population structure for PC calculation. Then uses pcair and pcrelate to leverage information from kinship(s) matrix(ices) generated from KING to adjust for relatedness in population. The first 200 PCs will be generated, if possible, as specified by the *v=200* parameter in the pcair call. <u>This will generate the PC table to be used for adjusting any PCs of interest in GENESIS association analysis calls and for plotting PC graphs.</u>

Next, the variable *pc_num*, specified in step 2 is used to adjust PCs by the number of ancestral populations specified in by this variable. Using the kinship matrices from KING, it tries to adjust the PCs by ancestral population PCs and relatedness by training on what KING and pcrelate consider an unrelated set until convergence is reached. <u>The pcrelate PC matrix will help generate the genetic relationships matrix.</u>

Warning: if **fullKin = TRUE** but the .kin file (within family relationships) does not contain any data, the logic is set so that it will generate the PCs only using the between family data from the .kin0 file as if fullKin = FALSE.

Step 6: Add all phenotypes from phenoFile and other covariates

- No changes to the code need to be made here.

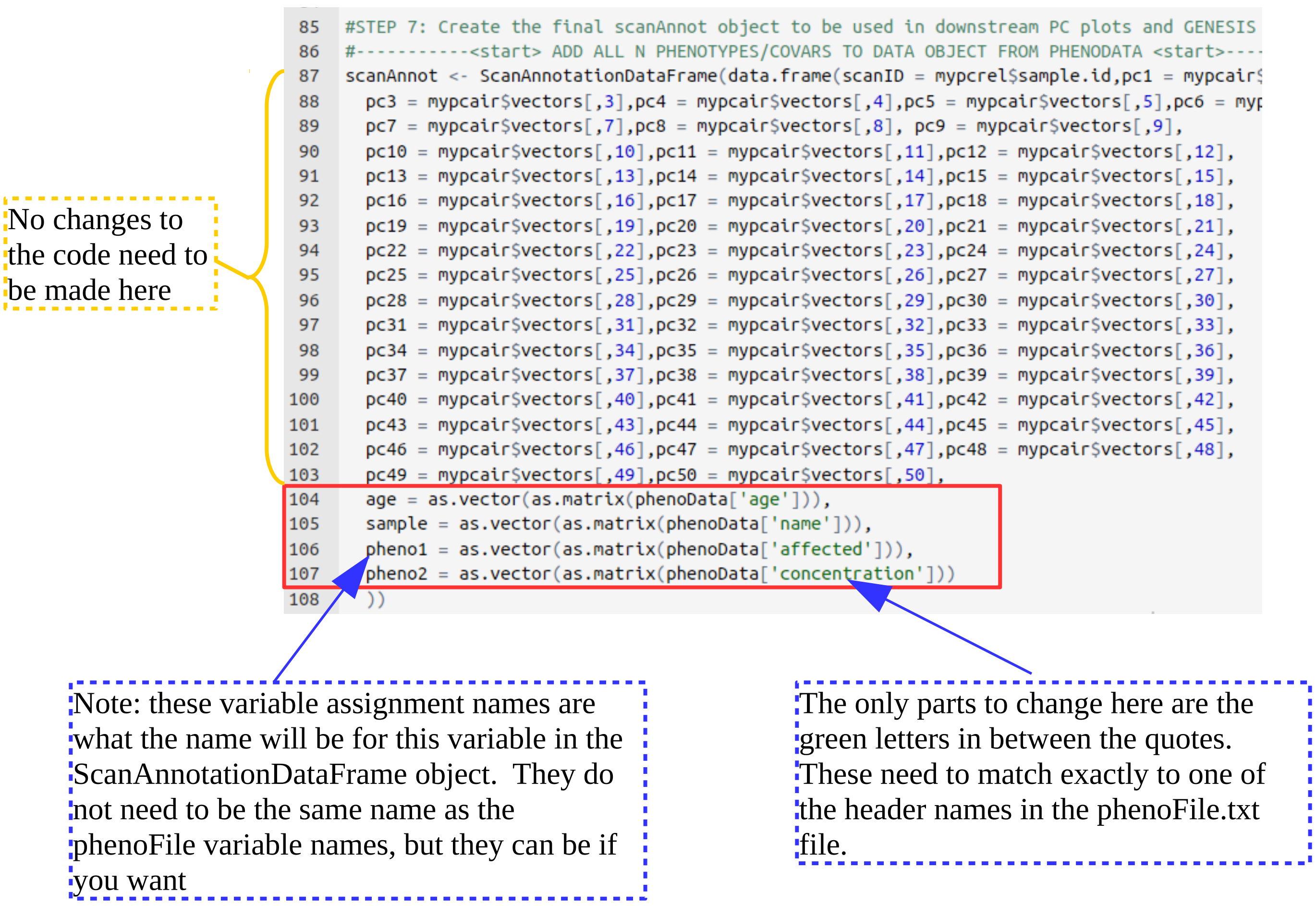
  Explanation of code:

  > Reads in the phenoFile.txt and stores it as a data frame with the headers matching what is listed in phenoFile.txt. It stores the data frame in a variable called *phenoData*. All phenotypes of interests and covariates to be used in downstream association analysis should be stored in this data frame.

Step 7: Create the final scanAnnot Dataframe to be used in downstream PC plots and GENESIS association analysis

- Change the parts boxed in red below to match which ever values you want to add to the data frame from your phenoFile.txt. In the example below, we had 4 values listed in the dummy_phenoFile: name, affected, concentration, and age. The script automatically stored these values in step 6 in the variable phenoData. Therefore, each variable added to ScanAnnotationDataFrame() can just use the same code convention:

```
nameInScanAnnot = as.vector(as.matrix(phenoData[`myVariableInPhenoFile’]))
```



```
#STEP 7: Create the final scanAnnot object to be used in downstream PC plots and GENESIS
#-----------<start> ADD ALL N PHENOTYPES/COVARS TO DATA OBJECT FROM PHENODATA <start>----
scanAnnot <- ScanAnnotationDataFrame(data.frame(scanID = mypcrel$sample.id,pc1 = mypcair$
  pc3 = mypcair$vectors[,3],pc4 = mypcair$vectors[,4],pc5 = mypcair$vectors[,5],pc6 = myp
  pc7 = mypcair$vectors[,7],pc8 = mypcair$vectors[,8], pc9 = mypcair$vectors[,9],
  pc10 = mypcair$vectors[,10],pc11 = mypcair$vectors[,11],pc12 = mypcair$vectors[,12],
  pc13 = mypcair$vectors[,13],pc14 = mypcair$vectors[,14],pc15 = mypcair$vectors[,15],
  pc16 = mypcair$vectors[,16],pc17 = mypcair$vectors[,17],pc18 = mypcair$vectors[,18],
  pc19 = mypcair$vectors[,19],pc20 = mypcair$vectors[,20],pc21 = mypcair$vectors[,21],
  pc22 = mypcair$vectors[,22],pc23 = mypcair$vectors[,23],pc24 = mypcair$vectors[,24],
  pc25 = mypcair$vectors[,25],pc26 = mypcair$vectors[,26],pc27 = mypcair$vectors[,27],
  pc28 = mypcair$vectors[,28],pc29 = mypcair$vectors[,29],pc30 = mypcair$vectors[,30],
  pc31 = mypcair$vectors[,31],pc32 = mypcair$vectors[,32],pc33 = mypcair$vectors[,33],
  pc34 = mypcair$vectors[,34],pc35 = mypcair$vectors[,35],pc36 = mypcair$vectors[,36],
  pc37 = mypcair$vectors[,37],pc38 = mypcair$vectors[,38],pc39 = mypcair$vectors[,39],
  pc40 = mypcair$vectors[,40],pc41 = mypcair$vectors[,41],pc42 = mypcair$vectors[,42],
  pc43 = mypcair$vectors[,43],pc44 = mypcair$vectors[,44],pc45 = mypcair$vectors[,45],
  pc46 = mypcair$vectors[,46],pc47 = mypcair$vectors[,47],pc48 = mypcair$vectors[,48],
  pc49 = mypcair$vectors[,49],pc50 = mypcair$vectors[,50],
  age = as.vector(as.matrix(phenoData['age'])),
  sample = as.vector(as.matrix(phenoData['name'])),
  pheno1 = as.vector(as.matrix(phenoData['affected'])),
  pheno2 = as.vector(as.matrix(phenoData['concentration']))
  ))
```

No changes to the code need to be made here

Note: these variable assignment names are what the name will be for this variable in the ScanAnnotationDataFrame object. They do not need to be the same name as the phenoFile variable names, but they can be if you want

The only parts to change here are the green letters in between the quotes. These need to match exactly to one of the header names in the phenoFile.txt file.

Step 8: Create the Genetic Relationships Matrix

- No changes to the code need to be made here.

  Explanation of code:

  Uses the pcrelate function called pcrelateMakeGRM() to create a genetic relationships matrix using the pcs generated from pcrelate in step 5. This is a different set of PCs from those in the scanAnnotationDataFrame(), which are using the PCs generated from pcair.

Step 9: Save data object and finish

- No changes to the code need to be made here.

  Explanation of code:

  Creates and saves three data objects: the scanAnnotationDataFrame, the genetic relationship matrix, and the pcair values (all 200 PCs, if available). This can be opened in R by using the load() function