Module 2. QC of the core subsamples

The steps in the “QC of the core subsamples” are intended to be executed on each core subsample. As an example, the steps below use the European subsample as we will prioritize the European subsamples because they account for the majority of the MoBa cohort. Once you have finished QC-ing the European subsample, please re-run all the steps on other subsamples in your batch. Please keep the file names the same, except for the population indicator. We use “eur” to indicate a European subsample, please use “afr” and “asian” as indicators of African and Asian subsamples (as seen in step 5.l of “identify core samples”). When you read “record” in the instructions below, it means to fill out the google sheet named “MoBa\_QC\_numbers” that is created to keep track of the QC numbers for the report.

# Quality Control (QC) steps

Load PLINK module

Follow the same setup as described in module 1 step-by-step document , e.g. set **GITHUB=/cluster/projects/p697/github/norment/moba\_qc\_imputation** if you work on machine with /cluster access (e.g. p697-submit), or **GITHUB=/tsd/p697/data/durable/s3-api/github/norment/moba\_qc\_imputation** if you work on a machine without /cluster access.

Unless otherwise specified, all commands are supposed to be run in your working directory (named with your initials) in DATA/DURABLE (see folder-structure-moba-2020.pdf)

All plots produced in this module should be copied to /tsd/p697/data/durable/projects/moba\_qc\_imputation/export/Module\_II\_Plots folder when indicated. “Record” in the instructions below refers to recording the numbers in MoBa\_QC\_numbers spreadsheet on Google drive, when indicated. Please contact Elizabeth if you have any doubts/questions about the process.

# FIRST ROUND

## Basic QC

### MAF

* + 1. Run the following command to remove variants with MAF <0.5%:  
       plink --bfile original-initials-eur --maf 0.005 --make-bed --out original-initials-eur-common
    2. Please record the number of SNPs removed in this step.
    3. Make a list of IDs of rare SNPs removed:  
       ./match.pl -f original-initials-eur-common.bim -g original-initials-eur.bim -k 2 -l 2 -v 1 | awk ‘$7==”-” {print $2,”rare, round 1”}’ > original-initials-eur-bad-snps.txt

### Call rates

* + 1. Histograms of missing rates
       1. Run the following command to identify the missing rates for variants and for individuals.  
          plink --bfile original-initials-eur-common --missing --out original-initials-eur-missing  
          This command will generate two files: one with extension of “.imiss” and another with extension of “.lmiss”. The “.imiss” file contains the missing rates per individual and the “.lmiss” file contains missing rates per SNP.
       2. Plot the missing rates in R  
          Use the script named “$GITHUB/lib/plot-missingness-histogram.R” (see the $GITHUB/lib/README file for more information about the script functionality and it’s input and output arguments). Run the script according to its manual.  
          The script will create three plots: one for individuals’ missingness (original-initials-eur-missing-rate-indiv.png) and two for the SNPs’ missingness (original-initials-eur-missing-rate-snps-all.png and original-initials-eur-missing-rate-snps-zoom.png).
          1. Example of how to run the plot-missingness-histogram.R script in your working directory (in terminal):  
             Rscript $GITHUB/lib/plot-missingness-histogram.R dataprefix “tag”  
             Where: dataprefix is the prefix of your .imiss and .lmiss files created in this step (specific to your batch and core subsample, i.e. original-initials-eur-missing).  
             tag (make sure it is written in quotation marks) consists of three arguments, first the name of your batch, second the name of the core subpopulation, and third the QC round.
       3. Please record the names of the plots you’ve created in MoBa\_QC\_numbers spreadsheet on google drive and copy the files with plots to the export folder for module II.
    2. Remove SNPs with call rate below 95%  
       plink --bfile original-initials-eur-common --geno 0.05 --make-bed --out original-initials-eur-95
    3. Please record the number of SNPs that failed this filter.
    4. Make a file with IDs of SNPs that were removed in this step using the following command:  
       cut -f2 original-initials-eur-common.bim original-initials-eur-95.bim | sort | uniq -c | awk '{if($1==1) print($2)}' > original-initials-eur-95-failed-snps.txt
    5. Remove SNPs with call rate below 98%:  
       plink --bfile original-initials-eur-95 --geno 0.02 --make-bed --out original-initials-eur-98
    6. Remove individuals and then SNPs with call rate below 98%:  
       plink --bfile original-initials-eur-98 --mind 0.02 --geno 0.02 --make-bed --out original-initials-eur-call-rates
    7. Please record the number of SNPs that fail the geno filter and the number of individuals who fail the mind filter.
    8. Create a file with the IDs of SNPs that failed this filter using the following command:  
       cut -f2 original-initials-eur-95.bim original-initials-eur-call-rates.bim | sort | uniq -c | awk '{if($1==1) print($2)}' > original-initials-eur-basic-qc-snp-call-rate-fail.txt
    9. Create a file with the IDs of individuals that failed this filter using the following command:  
       The list of individuals that failed this filter is stored in original-initials-eur-call-rates.irem created by PLINK.

### HWE

* + 1. HWE test, run the following command to remove SNPs not in HWE (with p<1.00E-06):  
       plink --bfile original-initials-eur-call-rates --hwe 0.000001 --make-bed --out original-initials-eur-basic-qc
    2. Please record the number of SNPs that fail the HWE filter.
    3. Create a file with the IDs of SNPs that failed.  
       cut -f2 original-initials-eur-call-rates.bim original-initials-eur-basic-qc.bim | sort | uniq -c | awk '{if($1==1) print($2)}' > original-initials-eur-basic-qc-snp-hwe-fail.txt

### Heterozygosity

* + 1. Estimate heterozygosity and missingness with the following PLINK command  
       plink --bfile original-initials-eur-basic-qc --chr 1-22 --het --missing --out original-initials-eur-common
    2. Plot the data and make a list of outliers based on being outside the +/- 3 standard deviations of the sample mean.
       1. Use the script “$GITHUB/lib/plot-heterozygosity-common.R” to run in R. (see the $GITHUB/lib/README.md file for more information about the script functionality and it’s input and output arguments).
       2. Usage: Rscript $GITHUB/lib/plot-heterozygosity-common.R dataprefix “tag”  
          Where: dataprefix - prefix of the outputs from PLINK command to estimate heterozygosity and missingness  
          “tag” - “tag population, round 2”, please remember to use the tag from “plot-PLINK” file in “resources” folder.
    3. Please record the number of individuals who are the outliers, you can find the number by using the following command:  
       tail -n +2 original-initials-eur-common-het-fail.txt | wc -l
    4. Remove heterozygosity outliers:  
       plink --bfile original-initials-eur-basic-qc --remove original-initials-eur-common-het-fail.txt --make-bed --out original-initials-eur-het
    5. Add the list of individuals to those who have already been removed.

## Sex check

* 1. Make sure the X chromosome pseudo-autosomal region is coded as a separate XY chromosome using the following command:  
     plink --bfile original-initials-eur-het --split-x b37 --make-bed --out original-initials-eur-pseudo
  2. Run sexcheck in PLINK.
     1. If the pseudo-autosomal region was already coded use the following command:  
        plink --bfile original-initials-eur-het --check-sex --out original-initials-eur-sexcheck-1
     2. If the pseudo-autosomal region was not coded use the following command:  
        plink --bfile original-initials-eur-pseudo --check-sex --out original-initials-eur-sexcheck-1
  3. Make a plot to get an overview of the reported sex in the batch
     1. Get missingness for chromosome X
        1. If the pseudo-autosomal region was already coded use the following command:  
           plink --bfile original-initials-eur-het --chr 23 --missing --out original-initials-eur-chr23-miss
        2. If the pseudo-autosomal region was not coded use the following command:  
           plink --bfile original-initials-eur-pseudo --chr 23 --missing --out original-initials-eur-chr23-miss
     2. Add chromosome X missingness to the sex check file  
        ./match.pl -f original-initials-eur-chr23-miss.imiss -g original-initials-eur-sexcheck-1.sexcheck -k 2 -l 2 -v 6 > original-initials-eur-1-chr23-plot.txt
     3. Run the $GITHUB/lib/plot-sex.R script. Please see README-for-plot-sex for usage information. Usage: Rscript $GITHUB/lib/plot-sex.R input tag legendpos output  
        Where: input - the name of the file containing F and missingness data for X chromosome  
        tag - part of the title to be given to the plot, it should reflect the tag of the batch from plot-PLINK file and the subpopulation/round of the QC  
        legendpos - the position of legend (use: topleft, topright, bottomleft, bottomright)  
        output - the name of the output file to be created
        1. Please specify the:  
           input - original-initials-eur-1-chr23-plot.txt  
           title - “tag.from.plot-PLINK.file EUR, round 1”  
           output - original-initials-eur-1-sex-plot.png  
           Example: Rscript $GITHUB/lib/plot-sex.R original-initials-eur-1-chr23-plot.txt “tag EUR, round 1” topleft original-initials-eur-1-sex-plot.png
     4. Please post the plot on slack and, if everything is OK, copy it to /tsd/p697/data/durable/projects/moba\_qc\_imputation/export/Module\_II\_Plots folder.
  4. Identify individuals with problematic sex assignment.  
     awk '$3!=0 && $5=="PROBLEM" {print $0}' original-initials-eur-sexcheck-1.sexcheck > original-initials-eur-bad-sex-1.txt
  5. Record the number of individuals with problematic sex assignment.  
     wc -l original-initials-eur-bad-sex-1.txt
  6. Identify individuals with erroneous sex assignment  
     awk '$3==1 && $6<0.5 || $3==2 && $6>0.5 {print $0}' original-initials-eur-bad-sex-1.txt > original-initials-eur-erroneous-sex-1.txt
  7. Record the number of individuals with erroneous sex assignment.  
     wc -l original-initials-eur-erroneous-sex-1.txt
  8. Remove individuals with erroneous sex assignment:  
     plink --bfile original-initials-eur-het --remove original-initials-eur-erroneous-sex-1.txt --make-bed --out original-initials-eur-sex-1
  9. Add the IDs of individuals removed to the list of those who have already been removed.

## Duplicates

QC of duplicates will only be done in the first round of this module for batches that have within batch duplicates. If your batch does not have duplicates proceed to step 4.To determine if your batch has within-batch duplicates look refer to the file /tsd/p697/data/durable/projects/moba\_qc\_imputation/resources/duplicates.xlsx.

### Identify real duplicates

* + 1. Prune the data using the following command:  
       plink --bfile original-initials-eur-sex-1 --indep-pairwise 3000 1500 0.1 --out original-initials-eur-prune
    2. Copy the relevant list of duplicates (the lists are by genotype array: hce\_duplicates.txt, omni\_duplicates.txt, and gsa\_duplicates.txt) from the “resources folder” on DATA/DURABLE to your working directory.
    3. Run IBD calculation in PLINK for expected duplicates (example below is for batches genotyped using the hce array).  
       plink --bfile original-initials-eur-sex-1 --extract original-initials-eur-prune.prune.in --keep hce\_duplicates.txt --genome --out original-initials-eur-ibd
    4. Examine the log file. If no individuals remain after “--keep”, move on to step 4.
    5. Run the following command to keep only real duplicates (i.e., those that share at least 98% IBD):  
       awk '$10>=0.98 {print $0}' original-initials-eur-ibd.genome > original-initials-eur-good-dup.txt
    6. For the ‘real’ duplicate pairs split the data into two sets using the following commands:  
       awk ‘{print $1,$2}’ original-initials-eur-good-dup.txt | sort | uniq > original-initials-eur-good-dup-1.txt  
       awk ‘{print $3,$4}’ original-initials-eur-good-dup.txt | sort | uniq > original-initials-eur-good-dup-2.txt
    7. For known triplicates identify the individual (IID) that is in both the original-initials-eur-good-dup-1.txt and original-initials-eur-good-dup-2.txt file. The hce\_duplicates\_wide.txt, omni\_duplicates\_wide.txt, or gsa\_duplicates\_wide.txt file in the resources folder should be used to help with this. Each row of the duplicates\_wide files contains a duplicate/triplicates pair.
       1. Create a file named original-initials-eur-good-dup-3.txt that contains the individual in both the original-initials-eur-good-dup-1.txt and original-initials-eur-good-dup-2.txt files.  
          awk ‘NR==FNR{A[$2];next}$2 in A’ original-initials-eur-good-dup-1.txt original-initials-eur-good-dup-2.txt > original-initials-eur-good-dup-3.txt
       2. Remove the individual from both the original-initials-eur-good-dup-1.txt and original-initials-eur-good-dup-2.txt files.  
          grep -vwE -f original-initials-eur-good-dup-3.txt original-initials-eur-good-dup-1.txt > original-initials-eur-good-dup-1.txt\_temp  
          grep -vwE -f original-initials-eur-good-dup-3.txt original-initials-eur-good-dup-2.txt > original-initials-eur-good-dup-2.txt\_temp  
          mv original-initials-eur-good-dup-1.txt\_temp original-initials-eur-good-dup-1.txt  
          mv original-initials-eur-good-dup-2.txt\_temp original-initials-eur-good-dup-2.txt

### SNP concordance analysis

* + 1. Create PLINK bfiles for the sets of real duplicates/triplicates using the following commands:   
       plink --bfile original-initials-eur-sex-1 --keep original-initials-eur-good-dup-1.txt --make-bed --out original-initials-eur-good-dup-1  
       plink --bfile original-initials-eur-sex-1 --keep original-initials-eur-good-dup-2.txt --make-bed --out original-initials-eur-good-dup-2  
       plink --bfile original-initials-eur-sex-1 --keep original-initials-eur-good-dup-3.txt --make-bed --out original-initials-eur-good-dup-3
       1. Check that the number of individuals in original-initials-eur-good-dup-1.fam and original-initials-eur-good-dup-2.fam is the same. The results of the following commands should be the same:  
          wc -l original-initials-eur-good-dup-1.fam  
          wc -l original-initials-eur-good-dup-2.fam  
          wc -l original-initials-eur-good-dup-3.fam
    2. Rename the duplicates in one set of PLINK bfiles using the following commands:  
       awk ‘{print $3, $4, $1, $2}’ original-initials-eur-good-dup.txt > original-initials-eur-good-dup-id.txt  
       plink --bfile original-initials-eur-good-dup-2 --update-ids original-initials-eur-good-dup-id.txt --make-bed --out original-initials-eur-good-dup-id
    3. Rename the triplicates.
       1. Use the hce\_duplicates\_v12.txt file in the resources folder to make original-initials-eur-good-trip1-id.txt and original-initials-eur-good-trip2-id.txt files with the first two columns containing the FID and IID of the individual(s) in the original-initials-eur-good-dup-3.txt file and the last two columns containing the FID and IID of the triplicate individual(s) in the original-initials-eur-good-trip1-id.txt and original-initials-eur-good-trip2-id.txt files, respectively.  
          grep -wE -f hce\_triplicates\_v12.txt original-initials-eur-good-dup-1.txt > original-initials-eur-good-trip1-id.txt\_temp  
          grep -wE -f hce\_triplicates\_v12.txt original-initials-eur-good-dup-2.txt > original-initials-eur-good-trip2-id.txt\_temp  
          paste original-initials-eur-good-dup-3.txt original-initials-eur-good-trip1-id.txt\_temp > original-initials-eur-good-trip1-id.txt  
          paste original-initials-eur-good-dup-3.txt original-initials-eur-good-trip2-id.txt\_temp > original-initials-eur-good-trip2-id.txt  
          rm original-initials-eur-good-trip1-id.txt\_temp  
          rm original-initials-eur-good-trip2-id.txt\_temp
       2. Run the plink commands to rename the triplicates.  
          plink --bfile original-initials-eur-good-dup-3 --update-ids original-initials-eur-good-trip1-id.txt --make-bed --out original-initials-eur-good-trip1-id  
          plink --bfile original-initials-eur-good-dup-3 --update-ids original-initials-eur-good-trip2-id.txt --make-bed --out original-initials-eur-good-trip2-id
    4. Identify SNPs that are discordant in the real duplicates using the following command:  
       plink --bfile original-initials-eur-good-dup-1 --bmerge original-initials-eur-good-dup-id --merge-mode 7 --out original-initials-eur-dup-check
    5. Identify SNPs that are discordant in the real triplicates using the following commands:  
       plink --bfile original-initials-eur-good-dup-1 --bmerge original-initials-eur-good-trip1-id --merge-mode 7 --out original-initials-eur-trip1-check  
       plink --bfile original-initials-eur-good-dup-2 --bmerge original-initials-eur-good-trip2-id --merge-mode 7 --out original-initials-eur-trip2-check
    6. Create a list of discordant SNPs
       1. If you only had duplicates use the following command:  
          awk '{print $1}' original-initials-eur-dup-check.diff | sort -u > original-initials-eur-dup-snps-to-remove.txt
       2. If you had triplicates use the following commands:  
          cat original-initials-eur-dup-check.diff original-initials-eur-trip1-check.diff original-initials-eur-trip2-check.diff > original-initials-eur-trip-check.diff  
          awk '{print $1}' original-initials-eur-trip-check.diff | sort -u > original-initials-eur-dup-snps-to-remove.txt
    7. Record the number of discordant SNPs to be removed  
        wc -l original-initials-eur-dup-snps-to-remove.txt
    8. Remove the discordant SNPs using the following command:  
       plink --bfile original-initials-eur-sex-1 --exclude original-initials-eur-dup-snps-to-remove.txt --make-bed --out original-initials-eur-dup-clean
    9. Create a file with the IDs of SNPs that were removed.

### Remove one individual from each duplicate pair

* + 1. Get the call rates of individuals using the following command:  
       plink --bfile original-initials-eur-dup-clean --missing --out original-initials-eur-dup-clean-miss
    2. Copy the relevant duplicates file in wide format, where one row represents one duplicate/triplicate pair/trio, to your working directory (the lists are by genotype array: hce\_duplicates\_wide\_no\_r.txt, omni\_duplicates\_wide.txt, and gsa\_duplicates\_wide.txt) from the “resources folder” on DATA/DURABLE to your working directory.
    3. Add the missingness rates to the duplicates file in wide format using the following commands (hce\_duplicates\_wide\_no\_r.txt is used in the below example):  
       ./match.pl -f original-initials-eur-dup-clean-miss.imiss -g hce\_duplicates\_wide\_no\_r.txt -k 2 -l 1 -v 6 > original-initials-eur-dup-miss1  
       ./match.pl -f original-initials-eur-dup-clean-miss.imiss -g original-initials-eur-dup-miss1 -k 2 -l 2 -v 6 > original-initials-eur-dup-miss2  
       ./match.pl -f original-initials-eur-dup-clean-miss.imiss -g original-initials-eur-dup-miss2 -k 2 -l 3 -v 6 > original-initials-eur-dup-miss3  
       rm original-initials-eur-dup-miss1  
       rm original-initials-eur-dup-miss2
    4. Identify duplicates/triplicates with lower call rate than their pair(s) using the following commands:  
       awk '$6=="-" && $4!="-" && $5!="-" {print $0}' original-initials-eur-dup-miss3 | awk '{if($4>$5) print $1; else print $2}' > original-initials-eur-max  
       awk '$5=="-" && $4!="-" && $6!="-" {print $0}' original-initials-eur-dup-miss3 | awk '{if($4>$6) print $1; else print $3}' >> original-initials-eur-max  
       awk '$4=="-" && $5!="-" && $6!="-" {print $0}' original-initials-eur-dup-miss3 | awk '{if($5>$6) print $2; else print $3}' >> original-initials-eur-max  
       awk '$4!="-" && $5!="-" && $6!="-" {print $0}' original-initials-eur-dup-miss3 | awk '{if($4<=$5 && $4<=$6) print $2"\n"$3; else if($5<=$4 && $5<=$6) print $1"\n"$3; else print $1"\n"$2}' >> original-initials-eur-max
    5. Create a list of individuals to remove   
       ./match.pl -f original-initials-eur-max -g hce\_duplicates.txt -k 1 -l 2 -v 1 | awk '$3!="-" {print $1,$2}' > original-initials-eur-multi-bad
    6. Remove the individuals from the duplicate/triplicate pairs to remove using the following command:  
       plink --bfile original-initials-eur-dup-clean --remove original-initials-eur-multi-bad --make-bed --out original-initials-eur-multi-clean
    7. Record how many individuals were removed.
    8. Add the IDs of individuals removed to the list of previously removed individuals:

## Unlinkable individuals

* 1. Copy the file named “unlinkable\_IDs.txt” from the resources folder to your working directory. This file contains IDs of individuals for whom linkage information was not obtained from Central MoBa.
  2. Remove the individuals without linkage information
     1. If your batch had duplicates use the following command:  
        plink --bfile original-initials-eur-multi-clean --remove unlinkable\_IDs.txt --make-bed --out original-initials-eur-linked-only
     2. If your batch did not have duplicates, then use the following command:  
        plink --bfile original-initials-eur-sex-1 --remove unlinkable\_IDs.txt --make-bed --out original-initials-eur-linked-only
  3. Please record how many individuals were removed.

## Pedigree build and known relatedness

This can be a memory intensive step particularly if you have a large batch, therefore, it is recommended to run "king" directly on the p697-appn-norment01 machine or on Colossus. The software directory has multiple versions of "king", use the latest version: king225. NB king225rhel6 can be used on TSD login nodes (but only if submit nodes and app node are unavailable); king225rhel6 binary will crash on p697-submit nodes and p697-appn-norment01. Finally, king225\_patch1 is a customly modified "king" program which produces a smaller output file in "king --ibs". Only individuals with kinship above 0.025 are reported.

### KING pedigree build and relatedness.

* + 1. Copy the age.txt file from “resources” folder to your working directory on DATA/DURABLE.
    2. Rename the file so it has the same prefix as the PLINK bfiles using the following command:  
       mv age.txt original-initials-eur-linked-only.cov
    3. If you are running KING on the p697-appn-norment01 machine use the following command:  
       /cluster/projects/p697/projects/moba\_qc\_imputation/software/king225 -b original-initials-eur-linked-only.bed --related --ibs --build --degree 2 --rplot --prefix original-initials-eur-king-1 > original-initials-eur-king-1-slurm.txt
    4. If you are running KING on Colossus:
       1. Create a folder named with your initials on /cluster/projects/p697/projects/moba\_qc\_imputation (please see “folder-structure-moba-2020.pdf” for reference). It will be your working directory on Colossus. Please use it every time you run things on Colossus.
       2. Copy original-initials-eur-linked-only.bed, original-initials-eur-linked-only.bim, original-initials-eur-linked-only.fam, and original-initials-eur-linked-only.cov files to your working directory on Colossus.
       3. Once in your working directory on Colossus, define “fin” and “fout” environmental variables as shown below, adjusting the names to reflect the names of your files (e.g. “original-initials” needs to become “moba12good-ec”), and then submit the job to Colossus:  
          export fin=original-initials-eur-linked-only  
          export fout=original-initials-eur-king-1  
          sbatch $GITHUB/jobs/KING.job
       4. When you submit the job, please note the job number. Once the job is finished, rename the slurm output file using the following command:  
          mv slurm-your.jobnumber.out original-initials-eur-king-1-slurm.txt
       5. Move KING output files to your working directory on DATA/DURABLE and delete the input files used for this step from your working directory on Colossus. These can be achieved with the following commands:  
          rm original-initials-eur-linked-only.\*  
          mv original-initials-eur-king-1\*.\* /tsd/p697/data/durable/projects/moba\_qc\_imputation/initials
       6. Go back to your working directory on DATA/DURABLE and continue working in that directory for the rest of this step.
    5. Please check in the slurm file that age was taken into account when running the build section of KING. Please let Elizabeth know if age was not taken into account.

### Update pedigree according to KING output:

* + 1. Check that the IID, PID, and MID in the original-initials-eur-king-1updateids.txt or original-initials-eur-king-1updateparents.txt files have not been changed from SENTRIX format. If the IIDs have been changed from the SENTRIX format use the following commands to convert back to SENTRIX format. Otherwise proceed to step 5.5.
       1. Command to convert the updateids file:  
          awk '{print $1,$2,$3,$2}' original-initials-eur-king-1updateids.txt > original-initials-eur-king-1updateids.txt-sentrix
       2. Command to convert the updateparents file:  
          R  
          library(tidyr)  
          update <- read.table('original-initials-eur-king-1updateparents.txt',h=F)  
          update$Order <- 1:nrow(update)  
          update\_IID <- update[grep("->", update$V2),]  
          no\_update\_IID <- update[!update$Order %in% update\_IID$Order,]  
          update\_IID <- separate(update\_IID, V2, into=c(NA,"V2"), sep="->", remove=T)  
          update\_IID <- update\_IID[,c(1,3:9)]  
          update <- rbind(update\_IID, no\_update\_IID)  
          rm(update\_IID, no\_update\_IID)  
          update\_PID <- update[grep("->", update$V3),]  
          no\_update\_PID <- update[!update$Order %in% update\_PID$Order,]  
          update\_PID <- separate(update\_PID, V3, into=c(NA,"V3"), sep="->", remove=T)  
          update\_PID <- update\_PID[,c(1:2,4:9)]  
          update <- rbind(update\_PID, no\_update\_PID)  
          rm(update\_PID, no\_update\_PID)  
          update\_MID <- update[grep("->", update$V4),]  
          no\_update\_MID <- update[!update$Order %in% update\_MID$Order,]  
          update\_MID <- separate(update\_MID, V4, into=c(NA,"V4"), sep="->", remove=T)  
          update\_MID <- update\_MID[,c(1:3,5:9)]  
          update <- rbind(update\_MID, no\_update\_MID)  
          rm(update\_MID, no\_update\_MID)  
          update <- update[,c(1:7)]  
          write.table(update, 'original-initials-eur-king-1updateparents.txt-sentrix', row.names=F, col.names=F, sep='\t', quote=F)  
          q()
    2. Update family and individual IDs using the following commands:
       1. If the IID was not changed from SENTRIX format use:  
          plink --bfile original-initials-eur-linked-only --update-ids original-initials-eur-king-1updateids.txt --make-bed --out original-initials-eur-king-1-ids
       2. If the IID was changed from SENTRIX format use:  
          plink --bfile original-initials-eur-linked-only --update-ids original-initials-eur-king-1updateids.txt-sentrix --make-bed --out original-initials-eur-king-1-ids
    3. Update paternal and maternal IDs using the following commands:
       1. If the IID was not changed from SENTRIX format use:  
          plink --bfile original-initials-eur-king-1-ids --update-parents original-initials-eur-king-1updateparents.txt --make-bed --out original-initials-eur-king-1-parents
       2. If the IID was changed from SENTRIX format use:  
          plink --bfile original-initials-eur-king-1-ids --update-parents original-initials-eur-king-1updateparents.txt-sentrix --make-bed --out original-initials-eur-king-1-parents

### Identified relationships:

* + 1. Look at the original-initials-eur-king-1-slurm.txt file and record the number of various between and within family relationships that KING identified/inferred in comparison to MoBa (during the related analysis).

### YOB and Sex check

In the “resources” folder in DATA/DURABLE, there are files named “yob.txt” and “sex.txt” that contains the year of birth of MoBa participants (the three columns in the file are: FID, IID and year-of-birth) and the sex of MoBa participants (the three columns in the file are: FID, IID and sex), respectively. Copy these files to your working directory and run the commands below.

* + 1. YOB check  
       ./match.pl -f yob.txt -g original-initials-eur-king-1-parents.fam -k 2 -l 2 -v 3 > original-initials-eur-king-1-children-yob.txt  
       ./match.pl -f yob.txt -g original-initials-eur-king-1-children-yob.txt -k 2 -l 3 -v 3 > original-initials-eur-king-1-children-fathers-yob.txt  
       ./match.pl -f yob.txt -g original-initials-eur-king-1-children-fathers-yob.txt -k 2 -l 4 -v 3 > original-initials-eur-king-1-yob.txt  
       rm original-initials-eur-king-1-children-yob.txt original-initials-eur-king-1-children-fathers-yob.txt  
       awk ‘{if ($7<$8 || $7<$9) print $0, “PROBLEM”; else print $0, “OK”}’ original-initials-eur-king-1-yob.txt > original-initials-eur-king-1-yob-check.txt  
       awk ‘$10==”PROBLEM” {print $0}’ original-initials-eur-king-1-yob-check.txt > original-initials-eur-king-1-yob-problem.txt
    2. Sex check  
       ./match.pl -f original-initials-eur-king-1-parents.fam -g original-initials-eur-king-1-parents.fam -k 2 -l 3 -v 5 > original-initials-eur-king-1-father-sex.txt  
       ./match.pl -f original-initials-eur-king-1-parents.fam -g original-initials-eur-king-1-father-sex.txt -k 2 -l 4 -v 5 > original-initials-eur-king-1-sex.txt  
       rm original-initials-eur-king-1-father-sex.txt  
       awk ‘{if ($7==2 || $8==1) print $0, “PROBLEM”; else print $0, “OK”}’ original-initials-eur-king-1-sex.txt > original-initials-eur-king-1-sex-check.txt  
       awk ‘$10==”PROBLEM” {print $0}’ original-initials-eur-king-1-sex-check.txt > original-initials-eur-king-1-sex-problem.txt
    3. Check the number of problematic families. Use the following commands to get the number:  
       wc -l original-initials-eur-king-1-yob-problem.txt  
       wc -l original-initials-eur-king-1-sex-problem.txt

### Examine the relationships within families (kin file)

* + 1. Identify any instances where the inferred relationships do not match those reported in MoBa:  
       awk ‘$16>0 {print $0}’ original-initials-eur-king-1.kin > original-initials-eur-king-1.kin-errors  
       ./match.pl -f /tsd/p697/data/durable/projects/moba\_qc\_imputation/resources/genotyped\_pedigree.txt -g original-initials-eur-king-1.kin-errors -k 4 -l 2 -v 8 > original-initials-eur-king-1.kin-errors-role1  
       ./match.pl -f /tsd/p697/data/durable/projects/moba\_qc\_imputation/resources/genotyped\_pedigree.txt -g original-initials-eur-king-1.kin-errors-role1 -k 4 -l 3 -v 8 > original-initials-eur-king-1.kin-errors  
       rm original-initials-eur-king-1.kin-errors-role1  
       wc -l original-initials-eur-king-1.kin-errors
       1. Identify any inferred duplicates/MZ twins that were unexpected:  
          awk ‘$15==”Dup/MZ” {print $0}’ original-initials-eur-king-1.kin-errors > original-initials-eur-king-1.kin-errors-MZ  
          wc -l original-initials-eur-king-1.kin-errors-MZ
       2. Identify any inferred parent-offspring relationships that were unexpected:  
          awk ‘$15==”PO” {print $0}’ original-initials-eur-king-1.kin-errors > original-initials-eur-king-1.kin-errors-PO  
          wc -l original-initials-eur-king-1.kin-errors-PO
       3. Identify any inferred full siblings that were unexpected:  
          awk ‘$15==”FS” {print $0}’ original-initials-eur-king-1.kin-errors > original-initials-eur-king-1.kin-errors-FS  
          wc -l original-initials-eur-king-1.kin-errors-FS
       4. Identify any inferred second degree relatives that were unexpected:  
          awk ‘$15==”2nd” {print $0}’ original-initials-eur-king-1.kin-errors > original-initials-eur-king-1.kin-errors-2nd  
          wc -l original-initials-eur-king-1.kin-errors-2nd
       5. Identify any inferred third degree relatives that were unexpected:  
          awk ‘$15==”3rd” {print $0}’ original-initials-eur-king-1.kin-errors > original-initials-eur-king-1.kin-errors-3rd  
          wc -l original-initials-eur-king-1.kin-errors-3rd
       6. Identify any inferred fourth degree relatives that were unexpected:  
          awk ‘$15==”4th” {print $0}’ original-initials-eur-king-1.kin-errors > original-initials-eur-king-1.kin-errors-4th  
          wc -l original-initials-eur-king-1.kin-errors-4th
       7. Identify any inferred unrelated individuals that were unexpected:  
          awk ‘$15==”UN” {print $0}’ original-initials-eur-king-1.kin-errors > original-initials-eur-king-1.kin-errors-UN  
          wc -l original-initials-eur-king-1.kin-errors-UN

### Examine the relationships between families (kin0 file)

* + 1. Identify any instances where the inferred relationships do not match those reported in MoBa:  
       awk ‘{if ($14==”Dup/MZ” || $14==”PO” || $14==”FS”) print $0}’ original-initials-eur-king-1.kin0 > original-initials-eur-king-1.kin0-errors  
       ./match.pl -f /tsd/p697/data/durable/projects/moba\_qc\_imputation/resources/genotyped\_pedigree.txt -g original-initials-eur-king-1.kin0-errors -k 4 -l 2 -v 8 > original-initials-eur-king-1.kin0-errors-role1  
       ./match.pl -f /tsd/p697/data/durable/projects/moba\_qc\_imputation/resources/genotyped\_pedigree.txt -g original-initials-eur-king-1.kin0-errors-role1 -k 4 -l 4 -v 8 > original-initials-eur-king-1.kin0-errors  
       rm original-initials-eur-king-1.kin0-errors-role1  
       wc -l original-initials-eur-king-1.kin0-errors
       1. Identify any inferred duplicates/MZ twins that were unexpected:  
          awk ‘$14==”Dup/MZ” {print $0}’ original-initials-eur-king-1.kin0-errors > original-initials-eur-king-1.kin0-errors-MZ  
          wc -l original-initials-eur-king-1.kin0-errors-MZ
       2. Identify any inferred parent-offspring relationships that were unexpected:  
          awk ‘$14==”PO” {print $0}’ original-initials-eur-king-1.kin0-errors > original-initials-eur-king-1.kin0-errors-PO  
          wc -l original-initials-eur-king-1.kin0-errors-PO
       3. Identify any inferred full siblings that were unexpected:  
          awk ‘$14==”FS” {print $0}’ original-initials-eur-king-1.kin0-errors > original-initials-eur-king-1.kin0-errors-FS  
          wc -l original-initials-eur-king-1.kin0-errors-FS

### Plot the relationships

* + 1. Make histograms of the estimated kinship coefficients between families
       1. Create file that is smaller for reading into R  
          awk ‘{print $2, $4, $19}’ original-initials-eur-king-1.ibs0 > original-initials-eur-king-1.ibs0\_hist
       2. Run plot-kinship-histogram.R according to the instructions in $GITHUB/lib/README-for-plot-kinship-histograms.txt. Please specify the following: input: original-initials-eur-king-1.ibs0\_hist and output: original-initials-eur-king-1-hist  
          Example: Rscript $GITHUB/lib/plot-kinship-histogram.R m12good-ec-eur-king-1.ibs0\_hist m12good-ec-eur-king-1-hist
       3. Remove the file used to plot the kinship histogram  
          rm original-initials-eur-king-1.ibs0\_hist
       4. Please post the graph on slack, copy it to the export folder for module II, and record the name of the plot in the google sheet.
    2. Make plots to visualize the between and within family relationships as reported by MoBa vs inferred by KING
    3. Please run the create-relplot.sh script that uses the relplot R script from KING, and generates a png file with four or two merged subplots with optionally customized legend positions. Usage information is on github (<https://github.com/norment/moba_qc_imputation/tree/master/tools#create-relplotsh>).  
       Usage: sh $GITHUB/tools/create-relplot.sh r\_relplot tag [legendpos1 legendpos2 legendpos3 legendpos4]  
       Arguments:  
       r\_relplot - R script file for relplot from KING  
       tag - a tag of data shown in the titles of the plots  
       legendpos1 - legend position of plot 1: topleft, topright, bottomleft, bottomright  
       legendpos2 - legend position of plot 2: topleft, topright, bottomleft, bottomright  
       legendpos3 - legend position of plot 3: topleft, topright, bottomleft, bottomright  
       legendpos4 - legend position of plot 4: topleft, topright, bottomleft, bottomright  
       Example: sh create-relplot.sh rotterdam1-yc-eur-king-1\_relplot.R "Rotterdam1 EUR, round 1" topright bottomright topright bottomright
    4. Post the plots on slack, copy the plot file to the export folder for plots of Module II, and record the plot name in the google sheet for report production.

### Fix within and between family issues

Let Elizabeth know once you have completed the above steps and she will investigate the pedigree errors and create the update and remove files.

* + 1. Remove individuals with impossible relationships and update family and individual ids.  
       plink --bfile original-initials-eur-king-1-parents --remove original-initials-eur-king-1-unexpected-relationships.txt --update-ids original-initials-eur-king-1-fix-ids.txt --make-bed --out original-initials-eur-king-1-fix-ids
    2. Update paternal and maternal ids.  
       plink --bfile original-initials-eur-king-1-fix-ids --update-parents original-initials-eur-king-1-fix-parents.txt --make-bed --out original-initials-eur-king-1-fix-parents
    3. Please make sure the number of individuals in the remove, update-ids, and update-parents files matches the number of individuals removed or updated in the plink log outputs.

### Identify any pedigree issues

* + 1. Create the covariate file
       1. Copy the age.txt file to your working directory.
       2. Update the FID to the match those updated during the above KING analysis.  
          ./match.pl -f original-initials-eur-king-1-fix-parents.fam -g age.txt -k 2 -l 2 -v 1 | awk ‘$4!=”-” {print $4, $2, $3}’ > original-initials-eur-king-1-fix-parents.cov  
          sed -i ‘1 i FID IID Age’ original-initials-eur-king-1-fix-parents.cov
    2. Run KING build using the following command:  
       /cluster/projects/p697/projects/moba\_qc\_imputation/software/king225 -b original-initials-eur-king-1-fix-parents.bed --build --degree 2 --prefix original-initials-eur-king-1.5 > original-initials-eur-king-1.5-slurm.txt
    3. Identify any issues
       1. Go through the build log file and see if there were any pedigree errors reported.
       2. See if an update ids or parents file was produced.
    4. Let Elizabeth know if there were any issues and she will investigate what is going on.
    5. Remove individuals with impossible relationships and update family and individual ids.
       1. Remove individuals with impossible relationships and update family and individual ids.  
          plink --bfile original-initials-eur-king-1-fix-parents --remove original-initials-eur-king-1.5-unexpected-relationships.txt --update-ids original-initials-eur-king-1.5-fix-ids.txt --make-bed --out original-initials-eur-king-1.5-fix-ids
       2. Update paternal and maternal ids.  
          plink --bfile original-initials-eur-king-1.5-fix-ids --update-parents original-initials-eur-king-1.5-fix-parents.txt --make-bed --out original-initials-eur-king-1.5-fix-parents

## PCA with 1000 Genomes (1KG)

If you had no problematic families, please use plink files with original-initials-eur-king-1-parents prefix (as updated based on the information from KING). If you had problematic families, please use PLINK files created after you’ve corrected any errors there may have been. For example, plink files with original-initials-eur-king-1-fix-parents or original-initials-eur-king-1.5-fix-parents prefixes. The PLINK commands below exemplify a situation when there were no errors in KING inferences. Please adjust according to your data. If you have any questions about this, please let Elizabeth know on slack.

### Prune and remove long stretches of LD

* + 1. Prune the data  
       plink --bfile original-initials-eur-king-1.5-fix-parents --indep-pairwise 3000 1500 0.1 --out original-initials-eur-king-1.5-prune
       1. If needed, repeat the pruning, to remove residual LD, until there are about 100,000 SNPs left.
    2. Extract the set of pruned SNPs  
       plink --bfile original-initials-eur-king-1.5-fix-parents --extract original-initials-eur-king-1.5-prune.prune.in --make-bed --out original-initials-eur-king-1.5-pruned
    3. Long LD regions
       1. Copy the file containing the list of long LD regions (Build37) from the “resources” folder to your working directory (name of the file is “high-ld.txt”).
       2. Create high LD set  
          plink --bfile original-initials-eur-king-1.5-pruned --make-set high-ld.txt --write-set --out original-initials-eur-king-1.5-highld
       3. Exclude SNPs in high LD set  
          plink --bfile original-initials-eur-king-1.5-pruned --exclude original-initials-eur-king-1.5-highld.set --make-bed --out original-initials-eur-king-1.5-trimmed

### Identify SNPs overlapping with 1KG

* + 1. Copy the 1KG PLINK bfiles from the “resources” folder in DATA/DURABLE to your working directory (1kg.bed,1kg.bim and 1kg.fam); these files have the strand ambiguous SNPs already removed.
    2. Run the following commands to identify the overlapping SNPs  
       cut -f2 1kg.bim | sort -s > 1kg.bim.sorted  
       cut -f2 original-initials-eur-king-1.5-trimmed.bim | sort -s > original-initials-eur-king-1.5-trimmed.bim.sorted  
       join 1kg.bim.sorted original-initials-eur-king-1.5-trimmed.bim.sorted > original-initials-eur-1kg-snps.txt  
       rm -f 1kg.bim.sorted original-initials-eur-king-1.5-trimmed.bim.sorted
    3. Record the number of SNPs you have common in your batch and in 1KG  
       wc -l original-initials-eur-1kg-snps.txt

### Merge with the 1KG dataset

* + 1. Extract the overlapping SNPs
       1. In your batch:  
          plink --bfile original-initials-eur-king-1.5-trimmed --extract original-initials-eur-1kg-snps.txt --make-bed --out original-initials-eur-1kg-common
       2. In the 1KG dataset:  
          plink --bfile 1kg --extract original-initials-eur-1kg-snps.txt --make-bed --out 1kg-original-initials-eur-common
    2. Merge the bfiles  
       plink --bfile original-initials-eur-1kg-common --bmerge 1kg-original-initials-eur-common --make-bed --out original-initials-eur-1kg-merged
       1. If you have SNPs that have 3+ alleles, flip those alleles in 1kg data and merge again. To flip, run the following command:  
          plink --bfile 1kg-original-initials-eur-common --flip original-initials-eur-1kg-merged-merge.missnp --make-bed --out 1kg-original-initials-eur-flip  
          To merge again, run the following command:  
          plink --bfile original-initials-eur-1kg-common --bmerge 1kg-original-initials-eur-flip --make-bed --out original-initials-eur-1kg-second-merged
       2. If you still have SNPs with 3+ alleles after merging with flipped data, remove those SNPs from both 1KG and MoBa data. To remove the SNPs with 3+ alleles from MoBa and from 1kg, run the following commands:  
          plink --bfile original-initials-eur-1kg-common --exclude original-initials-eur-1kg-second-merged-merge.missnp --make-bed --out original-initials-eur-1kg-clean  
          plink --bfile 1kg-original-initials-eur-flip --exclude original-initials-eur-1kg-second-merged-merge.missnp --make-bed --out 1kg-original-initials-eur-clean  
          To merge again, run the following command:  
          plink --bfile original-initials-eur-1kg-clean --bmerge 1kg-original-initials-eur-clean --make-bed --out original-initials-eur-1kg-clean-merged
    3. Record how many SNPs will be used for PCA:  
       wc -l original-initials-eur-1kg-clean-merged.bim

### PCA

* + 1. Copy the populations.txt file from “resources” folder on DATA/DURABLE to your working directory. populations.txt is a text file containing the population, based on which PLINK will calculate the main PCs, in this case it will be just one word “parent”).
    2. Create the original-initials-fam-populations.txt file - a text file with 3 columns: family ID, individual ID and so-called population, in this case it will “parent” for unrelated individuals and “child” for related individuals. To create this file, do the following:  
       awk ‘{if($3==0 && $4==0) print $1,$2,”parent”; else print $1,$2,”child”}’ original-initials-eur-1kg-merged.fam > original-initials-fam-populations.txt
       1. If you did the flip, use original-initials-eur-1kg-second-merged.fam
       2. If you had to remove SNPs after the flip, use original-initials-eur-1kg-clean-merged.fam
    3. Run the PCA  
       plink --bfile original-initials-eur-1kg-merged --pca --within original-initials-fam-populations.txt --pca-clusters populations.txt --out original-initials-eur-1kg-pca
       1. If your data had SNPs with 3+ alleles and you did flip+merge steps, use the original-initials-eur-1kg-second-merged bfiles
       2. If your data had SNPs with 3+ alleles after flip+merge steps and you did remove+merge steps, then use the original-initials-eur-1kg-clean-merged bfiles
       3. If your batch is large and you need to run the PCA on Colossus, copy your PLINK input files into Colossus and use the PCA\_POP.job

### Plot PCs

* + 1. The individuals in 1kg.fam file are not ordered according to their population. But in order to assign colors during plotting, we need them to be in order. Thus, we will divide the file containing the PCs into “original-eur” and “1kg” portions, sort the “1kg” portion and combine the two together for plotting. This can be achieved with the following commands:  
       sort -k2 original-initials-eur-1kg-pca.eigenvec > original-initials-eur-1kg-pca-sorted   
       head -n a original-initials-eur-1kg-pca-sorted > original-initials-eur-pca-1  
       NB the “a” in the “head” command is the number of individuals in the “original” PLINK files that were merged with 1KG (it is the number of individuals in original-initials-eur-king-1.5-trimmed.fam file)  
       tail -n 1083 original-initials-eur-1kg-pca-sorted | sort -k2 > 1kg-initials-eur-pca-1  
       NB 1083 is the number of individuals in 1kg.fam  
       cat original-initials-eur-pca-1 1kg-initials-eur-pca-1 > original-initials-eur-1kg-pca
    2. Plot your batch with the 1KG dataset.  
       The code for plotting is located in $GITHUB/lib/plot-pca-with-1kg.R. Instructions are in $GITHUB/lib/README.md
       1. Please remember to use the “plot-PLINK” file in “resources” folder to assign the harmonized tag to the plots of your batch.  
          Please specify the “outprefix” as “original-initials-eur-1kg”.
       2. When you’ve made the plots, please post them on slack and place a copy in the export folder for Module II.
    3. If needed, select a cleaner subsample using the script named “$GITHUB/lib/select-subsamples-on-pca.R”. Please name the customfile as “original-initials-eur-pca-core-select-custom.txt” and the “outprefix” as “original-initials-eur-selection-2”.  
       For examples of <customfile> see in $GITHUB/config folder.
       1. Please note that here we select only EUR subsample (in other words, we tighten the previous selection). Thus, the customfile will have only EUR section. Your customfile may look like this:  
          eur\_zoom\_threshold: PC1> -0.01  
          eur\_draw\_threshold: PC1>0  
          eur\_legend\_position: bottomleft
          1. In this example, the selection is done based on PC1 only. Please add PC2 and/or other PCs as required.
    4. Once done, please copy the plots to /tsd/p697/data/durable/projects/moba\_qc\_imputation/export/Module\_II\_Plots folder.
    5. Please record the names of the plots in MoBa\_QC\_numbers sheet.

### Remove ancestry outliers

* + 1. Use the following command to remove PC outliers:  
       plink --bfile original-initials-eur-king-1.5-fix-parents --keep original-initials-eur-selection-2-core-subsample-eur.txt --make-bed --out original-initials-eur-1-keep
    2. Please record the number of individuals kept in MoBa\_QC\_numbers sheet (you can see the number in the log file of the above PLINK command).
    3. Add the IDs of individuals that were removed to the list already removed.
    4. If there are no outliers in this step, then take original-initials-eur-king-1.5-fix-parents.bed, original-initials-eur-king-1.5-fix-parents.bim and original-initials-eur-king-1.5-fix-parents.fam into step 7.

## PCA without 1KG

### Prune and remove long stretches of LD

* + 1. Prune the data
       1. If there were outliers  
          plink --bfile original-initials-eur-1-keep --indep-pairwise 3000 1500 0.1 --out original-initials-eur-1-keep-prune
       2. If there were no outliers in step 6, then run the following command:  
          plink --bfile original-initials-eur-king-1.5-fix-parents --indep-pairwise 3000 1500 0.1 --out original-initials-eur-1-keep-prune
    2. Check the number of remaining SNPs after pruning:  
       wc -l original-initials-eur-1-keep-prune.prune.in
       1. If the number is substantially larger than 100K, repeat the pruning until there are about 100K SNPs left.
    3. Remove long stretches of LD  
       plink --bfile original-initials-eur-1-keep --extract original-initials-eur-1-keep-prune.prune.in --make-set high-ld.txt --write-set --out original-initials-eur-1-keep-highld  
       plink --bfile original-initials-eur-1-keep --extract original-initials-eur-1-keep-prune.prune.in --exclude original-initials-eur-1-keep-highld.set --make-bed --out original-initials-eur-1-keep-trimmed
    4. Record the number of SNPs in original-initials-eur-1-keep-trimmed.bim   
       wc -l original-initials-eur-1-keep-trimmed.bim

### PCA

* + 1. Copy the populations.txt file from “resources” folder on DATA/DURABLE to your working directory. populations.txt is a text file containing the population, based on which PLINK will calculate the main PCs, in this case it will be just one word “parent”).
    2. Create the original-initials-fam-populations-eur.txt – a text file with 3 columns: family ID, individual ID and so-called population, in this case it will be “parent” for unrelated individuals and “child” for related individuals. To create this file, do the following:  
       awk ‘{if($3==0 && $4==0) print $1,$2,”parent”; else print $1,$2,”child”}’ original-initials-eur-1-keep-trimmed.fam > original-initials-eur-1-keep-populations.txt
    3. Run the PCA  
       plink --bfile original-initials-eur-1-keep-trimmed --pca --within original-initials-eur-1-keep-populations.txt --pca-clusters populations.txt --out original-initials-eur-1-keep-pca
       1. If your batch is large and you need to run the PCA on Colossus, copy your PLINK input files into Colossus and use the PCA\_POP.job

### Plot PCs

* + 1. Identify founders and non-founders  
       awk ‘{if($3==0 && $4==0) print $1,$2,”black”; else print $1,$2,”red” }’ original-initials-eur-1-keep.fam > original-initials-eur-1-keep-fam.txt  
       ./match.pl -f original-initials-eur-1-keep-fam.txt -g original-initials-eur-1-keep-pca.eigenvec -k 2 -l 2 -v 3 > original-initials-eur-1-keep-pca-fam.txt
    2. Create PC plots using the plot-batch-PCs.R script. See $GITHUB/lib/README-for-plot-batch-PCs for usage instructions. The input is original-initials-eur-1-keep-pca-fam.txt, title is “original EUR, round 1”, and output is original-initials-eur-1-pca.png
    3. Once done, please post the plots on slack, and copy the plots to the /tsd/p697/data/durable/projects/moba\_qc\_imputation/export/Module\_II\_Plots folder.
    4. Please record the names of the plots in MoBa\_QC\_numbers sheet.

### Remove PC outliers

* + 1. Select the individuals to keep (in this example, we select individuals with PC1 value more than zero; please adjust the awk command as needed). Please reach out to Elizabeth if you have any doubts/questions.   
       awk ‘$3>0 {print $1,$2}’ original-initials-eur-1-keep-pca.eigenvec > original-initials-eur-1-pca-keep.txt
    2. Create PC plots after outlier selection.
       1. Create a file of containing the individuals kept with founders and non-founder status, using the same threshold conditions as step 7.4.1.  
          awk ‘$3>0 {print $0}’ original-initials-eur-1-keep-pca-fam.txt > original-initials-eur-1-pca-fam-keep.txt
       2. Run the plot-batch-PCs.R script. See $GITHUB/lib/README-for-plot-batch-PCs for usage instructions. The input is original-initials-eur-1-pca-fam-keep.txt, title is “original EUR, round 1”, and output is original-initials-eur-1-pca-keep.png
       3. Once done, please post the plots on slack, and copy the plots to the /tsd/p697/data/durable/projects/moba\_qc\_imputation/export/Module\_II\_Plots folder.
       4. Please record the names of the plots in MoBa\_QC\_numbers sheet.
    3. Create PLINK files without the outliers  
       plink --bfile original-initials-eur-1-keep --keep original-initials-eur-1-pca-keep.txt --make-bed --out original-initials-eur-1-round-selection
    4. Please record the number of individuals kept in MoBa\_QC\_numbers sheet (you can see the number in the log file of the above PLINK command).
    5. If you had outliers, add their IDs to those who were removed previously:

# SECOND ROUND

Please let Elizabeth know on slack when you are entering round two of the QC.

## Basic QC

### Remove rare variants (MAF<0.5%)

* + 1. Run the following command to remove variants with MAF <0.5%:  
       plink --bfile original-initials-eur-1-round-selection --maf 0.005 --make-bed --out original-initials-eur-2-common
    2. Record the number of rare SNPs removed.
    3. Make a list of IDs of rare SNPs removed:   
       ./match.pl -f original-initials-eur-2-common.bim -g original-initials-eur-1-round-selection.bim -k 2 -l 2 -v 1 | awk ‘$7==”-” {print $2,”rare, round 2”}’ > original-initials-eur-2-bad-snps.txt

### Call rates

* + 1. Histograms of missing rates
       1. Run the following command to identify the missing rates for SNPs and for individuals:  
          plink --bfile original-initials-eur-2-common --missing --out original-initials-eur-2-common-missing  
          This command will generate two files: one with extension of “.imiss” and another with extension of “.lmiss”. The “.imiss” file contains the missing rates per individual and the “.lmiss” file contains missing rates per SNP.
       2. Plot the missing rates in R using the script named “$GITHUB/lib/plot-missingness-histogram.R” (see the $GITHUB/lib/README.md file for more information about the script functionality and it’s input and output arguments). Run the script according to its manual.
          1. Example of how to run the plot-missingness-histogram.R script in your working directory (in terminal):  
             Rscript $GITHUB/lib/plot-missingness-histogram.R dataprefix “tag”  
             Where: dataprefix is the prefix of your .imiss and .lmiss files created in this step (specific to your batch and core subsample, i.e. original-initials-eur-missing).  
             tag (make sure it is written in quotation marks) consists of three arguments, first the name of your batch, second the name of the core subpopulation, and third the QC round.  
             Example of the command:  
             Rscript $GITHUB/lib/plot-missingness-histogram.R original-initials-eur-2-common-missing “tag population, round 2”
       3. Please record the names of the plots you’ve created in MoBa\_QC\_numbers spreadsheet on google drive and copy the files with plots to the export folder for module II.
    2. Remove SNPs with call rate <95%  
       plink --bfile original-initials-eur-2-common --geno 0.05 --make-bed --out original-initials-eur-2-95
    3. Record the number of failed SNPs.
    4. Make a list with IDs of the SNPs failed at this step:  
       ./match.pl -f original-initials-eur-2-95.bim -g original-initials-eur-2-common.bim -k 2 -l 2 -v 1 | awk ‘$7==”-” {print $2,”call-rate-below-95, round 2”}’ >> original-initials-eur-2-bad-snps.txt
    5. Remove SNPs with call rate below 98%:  
       plink --bfile original-initials-eur-2-95 --geno 0.02 --make-bed --out original-initials-eur-2-98
    6. Remove SNPs and individuals whose call rate is below 98%  
       plink --bfile original-initials-eur-2-98 --geno 0.02 --mind 0.02 --make-bed --out original-initials-eur-2-call-rates
    7. Please record the number of SNPs and individuals failing the call rates in MoBa\_QC\_numbers sheet corresponding to your batch.
    8. Please make a file with IDs of the SNPs that failed the call rate in this step:  
       ./match.pl -f original-initials-eur-2-call-rates.bim -g original-initials-eur-2-95.bim -k 2 -l 2 -v 1| awk ‘$7==”-” {print $2,”call-rate-below-98, round 2”}’ >> original-initials-eur-2-bad-snps.txt
    9. Make a file with IDs of individuals to remove:   
       awk ‘{print $1,$2,”call-rate-below-98, round 2”}’ original-initials-eur-2-call-rates.irem > original-initials-eur-2-removed-individuals.txt

### HWE test

* + 1. HWE test, run the following command to remove SNPs not in HWE (with p<1.00E-06):  
       plink --bfile original-initials-eur-2-call-rates --hwe 0.000001 --make-bed --out original-initials-eur-2-basic-qc
    2. Please record the number of SNPs that fail the HWE filter.
    3. Create a file with the IDs of SNPs that failed:  
       ./match.pl -f original-initials-eur-2-basic-qc.bim -g original-initials-eur-2-call-rates.bim -k 2 -l 2 -v 1| awk ‘$7==”-” {print $2,”out-of-HWE, round 2”}’ >> original-initials-eur-2-bad-snps.txt

### Heterozygosity

* + 1. Estimate heterozygosity and missingness with the following PLINK command:  
       plink --bfile original-initials-eur-2-basic-qc --chr 1-22 --het --missing --out original-initials-eur-2-common-het-miss
    2. Plot the data and make a list of outliers based on being outside the +/- 3 standard deviations of the sample mean.
       1. Use the script “$GITHUB/lib/plot-heterozygosity-common.R” to run in the terminal. (see the $GITHUB/lib/README.md file for more information about the script functionality and it’s input and output arguments).
          1. Usage: Rscript $GITHUB/lib/plot-heterozygosity-common.R dataprefix “tag”  
             Where: dataprefix - prefix of the outputs from PLINK command to estimate heterozygosity and missingness  
             “tag” - “tag population, round 2”, please remember to use the tag from “plot-PLINK” file in “resources” folder.
    3. Please record the number of individuals who are the outliers, you can find the number by using the following command:  
       tail -n +2 original-initials-eur-2-common-het-miss-het-fail.txt | wc -l
    4. Remove heterozygosity outliers:  
       plink --bfile original-initials-eur-2-basic-qc --remove original-initials-eur-2-common-het-miss-het-fail.txt --make-bed --out original-initials-eur-2-het
    5. Add the IDs of outliers to those who were removed previously:  
       awk ‘{print $1,$2,”heterozygosity, round 2”}’ original-initials-eur-2-common-het-miss-het-fail.txt >> original-initials-eur-2-removed-individuals.txt

## SexCheck

* 1. Make sure the X chromosome pseudo-autosomal region is coded as a separate XY chromosome using the following command:  
     plink --bfile original-initials-eur-2-het --split-x b37 --make-bed --out original-initials-eur-2-pseudo
  2. Run sexcheck in PLINK
     1. If the pseudo-autosomal region was already coded use the following command:  
        plink --bfile original-initials-eur-2-het --check-sex --out original-initials-eur-2-sexcheck
     2. If the pseudo-autosomal region was not coded use the following command:  
        plink --bfile original-initials-eur-2-pseudo --check-sex --out original-initials-eur-2-sexcheck
  3. Make a plot to get an overview of the reported sex in the batch
     1. Get missingness for chromosome X
        1. If the pseudo-autosomal region was already coded use the following command:  
           plink --bfile original-initials-eur-2-het --chr 23 --missing --out original-initials-eur-2-chr23-miss
        2. If the pseudo-autosomal region was not coded use the following command:  
           plink --bfile original-initials-eur-2-pseudo --chr 23 --missing --out original-initials-eur-2-chr23-miss
     2. Add chromosome X missingness to the sex check file  
        ./match.pl -f original-initials-eur-2-chr23-miss.imiss -g original-initials-eur-2-sexcheck.sexcheck -k 2 -l 2 -v 6 > original-initials-eur-2-chr23-plot.txt
     3. Run the $GITHUB/lib/plot-sex.R script. Please see README-for-plot-sex for usage information. Usage: Rscript $GITHUB/lib/plot-sex.R input tag legendpos output  
        Where: input - the name of the file containing F and missingness data for X chromosome  
        tag - part of the title to be given to the plot, it should reflect the tag of the batch from plot-PLINK file and the subpopulation/round of the QC  
        legendpos - the position of legend (use: topleft, topright, bottomleft, bottomright)  
        output - the name of the output file to be created
        1. Please specify the:  
           input - original-initials-eur-2-chr23-plot.txt  
           title - “tag.from.plot-PLINK.file EUR, round 2”  
           output - original-initials-eur-2-sex-plot.png  
           Example: Rscript $GITHUB/lib/plot-sex.R original-initials-eur-2-chr23-plot.txt “tag EUR, round 2” topleft original-initials-eur-2-sex-plot.png
     4. Please post the plot on slack and, if everything is OK, copy it to /tsd/p697/data/durable/projects/moba\_qc\_imputation/export/Module\_II\_Plots folder.
  4. Identify individuals with problematic sex assignment:  
     awk '$3!=0 && $5=="PROBLEM" {print $0}' original-initials-eur-2-sexcheck.sexcheck > original-initials-eur-2-bad-sex.txt
  5. Record the number of individuals with problematic sex assignment.  
     wc -l original-initials-eur-2-bad-sex.txt
  6. Identify individuals with erroneous sex assignment  
     awk '$3==1 && $6<0.5 || $3==2 && $6>0.5 {print $0}' original-initials-eur-2-bad-sex.txt > original-initials-eur-2-erroneous-sex.txt
  7. Record how many individuals have erroneous sex assignment:  
     wc -l original-initials-eur-2-erroneous-sex.txt
  8. Remove individuals with erroneous sex assignment:  
     plink --bfile original-initials-eur-2-het --remove original-initials-eur-2-erroneous-sex.txt --make-bed --out original-initials-eur-2-sex
  9. Add the IDs of individuals removed to the list of those who have already been removed.  
     awk ‘{print $1,$2,”erroneous-sex-2”}’ original-initials-eur-2-erroneous-sex.txt >> original-initials-eur-2-removed-individuals.txt

## Pedigree build and known relatedness

This can be a memory intensive step particularly if you have a large batch, therefore, it is recommended to run "king" directly on the p697-appn-norment01 machine or on Colossus. The software directory has multiple versions of "king", use the latest version: king225. NB king225rhel6 can be used on TSD login nodes (but only if submit nodes and app node are unavailable); king225rhel6 binary will crash on p697-submit nodes and p697-appn-norment01. Finally, king225\_patch1 is a customly modified "king" program which produces a smaller output file in "king --ibs". Only individuals with kinship above 0.025 are reported.

### KING pedigree build and relatedness.

* + 1. Create the covariate file.
       1. Copy the age.txt file from “resources” folder to your working directory on DATA/DURABLE.
       2. Update the file to contain the FID from previous pedigree builds, and rename the file to the same prefix as the PLINK files you’ve copied to Colossus. For example, if the PLINK files you are using have prefix of original-initials-eur-2-sex, then rename the file to original-initials-eur-2-sex.cov. The command lines for this are:  
          ./match.pl -f original-initials-eur-2-sex.fam -g age.txt -k 2 -l 2 -v 1 | awk ‘$4!=”-” {print $4, $2, $3}’ > original-initials-eur-2-sex.cov  
          sed -i ‘1 i\FID IID Age’ original-initials-eur-2-sex.cov
    2. If you are running KING on the p697-appn-norment01 machine use the following command:  
       /cluster/projects/p697/projects/moba\_qc\_imputation/software/king225 -b original-initials-eur-2-sex.bed --related --ibs --build --degree 2 --rplot --prefix original-initials-eur-king-2 > original-initials-eur-king-2-slurm.txt
    3. If you are running KING on Colossus:
       1. Create a folder named with your initials on /cluster/projects/p697/projects/moba\_qc\_imputation (please see “folder-structure-moba-2020.pdf” for reference). It will be your working directory on Colossus. Please use it every time you run things on Colossus.
       2. Copy original-initials-eur-linked-only.bed, original-initials-eur-2-sex.bim, original-initials-eur-2-sex.fam, and original-initials-eur-2-sex.cov files to your working directory on Colossus.
       3. Once in your working directory on Colossus, define “fin” and “fout” environmental variables as shown below, adjusting the names to reflect the names of your files (e.g. “original-initials” needs to become “moba12good-ec”), and then submit the job to Colossus:  
          export fin=original-initials-eur-2-sex  
          export fout=original-initials-eur-king-2  
          sbatch $GITHUB/jobs/KING.job
       4. When you submit the job, please note the job number. Once the job is finished, rename the slurm output file using the following command:  
          mv slurm-your.jobnumber.out original-initials-eur-king-2-slurm.txt
       5. Move KING output files to your working directory on DATA/DURABLE and delete the input files used for this step from your working directory on Colossus. These can be achieved with the following commands:  
          rm original-initials-eur-2-sex.\*  
          mv original-initials-eur-king-2\*.\* /tsd/p697/data/durable/projects/moba\_qc\_imputation/initials
       6. Go back to your working directory on DATA/DURABLE and continue working in that directory for the rest of this step.
    4. Please check in the slurm file that age was taken into account when running the build section of KING. Please let Elizabeth know if age was not taken into account.

### Update pedigree according to KING output:

* + 1. Check that the IID, PID, and MID in the original-initials-eur-king-2updateids.txt or original-initials-eur-king-2updateparents.txt files have not been changed from SENTRIX format. If the IIDs have been changed from the SENTRIX format use the following commands to convert back to SENTRIX format. Otherwise proceed to step 10.2.2.
       1. Command to convert the updateids file:  
          awk '{print $1,$2,$3,$2}' original-initials-eur-king-2updateids.txt > original-initials-eur-king-2updateids.txt-sentrix
       2. Command to convert the updateparents file:  
          R  
          library(tidyr)  
          update <- read.table('original-initials-eur-king-2updateparents.txt',h=F)  
          update$Order <- 1:nrow(update)  
          update\_IID <- update[grep("->", update$V2),]  
          no\_update\_IID <- update[!update$Order %in% update\_IID$Order,]  
          update\_IID <- separate(update\_IID, V2, into=c(NA,"V2"), sep="->", remove=T)  
          update\_IID <- update\_IID[,c(1,3:9)]  
          update <- rbind(update\_IID, no\_update\_IID)  
          rm(update\_IID, no\_update\_IID)  
          update\_PID <- update[grep("->", update$V3),]  
          no\_update\_PID <- update[!update$Order %in% update\_PID$Order,]  
          update\_PID <- separate(update\_PID, V3, into=c(NA,"V3"), sep="->", remove=T)  
          update\_PID <- update\_PID[,c(1:2,4:9)]  
          update <- rbind(update\_PID, no\_update\_PID)  
          rm(update\_PID, no\_update\_PID)  
          update\_MID <- update[grep("->", update$V4),]  
          no\_update\_MID <- update[!update$Order %in% update\_MID$Order,]  
          update\_MID <- separate(update\_MID, V4, into=c(NA,"V4"), sep="->", remove=T)  
          update\_MID <- update\_MID[,c(1:3,5:9)]  
          update <- rbind(update\_MID, no\_update\_MID)  
          rm(update\_MID, no\_update\_MID)  
          update <- update[,c(1:7)]  
          write.table(update, 'original-initials-eur-king-2updateparents.txt-sentrix', row.names=F, col.names=F, sep='\t', quote=F)  
          q()
    2. Update family and individual IDs using the following commands:
       1. If the IID was not changed from SENTRIX format use:  
          plink --bfile original-initials-eur-2-sex --update-ids original-initials-eur-king-2updateids.txt --make-bed --out original-initials-eur-king-2-ids
       2. If the IID was changed from SENTRIX format use:  
          plink --bfile original-initials-eur-2-sex --update-ids original-initials-eur-king-2updateids.txt-sentrix --make-bed --out original-initials-eur-king-2-ids
    3. Update paternal and maternal IDs using the following commands:
       1. If the IID was not changed from SENTRIX format use:  
          plink --bfile original-initials-eur-king-2-ids --update-parents original-initials-eur-king-2updateparents.txt --make-bed --out original-initials-eur-king-2-parents
       2. If the IID was changed from SENTRIX format use:  
          plink --bfile original-initials-eur-king-2-ids --update-parents original-initials-eur-king-2updateparents.txt-sentrix --make-bed --out original-initials-eur-king-2-parents

### Identified relationships:

* + 1. Look at the original-initials-eur-king-2-slurm.txt file and record the number of various between and within family relationships that KING identified/inferred in comparison to MoBa (during the related analysis).

### YOB and Sex check

In the “resources” folder in DATA/DURABLE, there are files named “yob.txt” and “sex.txt” that contains the year of birth of MoBa participants (the three columns in the file are: FID, IID and year-of-birth) and the sex of MoBa participants (the three columns in the file are: FID, IID and sex), respectively. Copy these files to your working directory and run the commands below.

* + 1. YOB check  
       ./match.pl -f yob.txt -g original-initials-eur-king-2-parents.fam -k 2 -l 2 -v 3 > original-initials-eur-king-2-children-yob.txt  
       ./match.pl -f yob.txt -g original-initials-eur-king-2-children-yob.txt -k 2 -l 3 -v 3 > original-initials-eur-king-2-children-fathers-yob.txt  
       ./match.pl -f yob.txt -g original-initials-eur-king-2-children-fathers-yob.txt -k 2 -l 4 -v 3 > original-initials-eur-king-2-yob.txt  
       rm original-initials-eur-king-2-children-yob.txt original-initials-eur-king-2-children-fathers-yob.txt  
       awk ‘{if ($7<$8 || $7<$9) print $0, “PROBLEM”; else print $0, “OK”}’ original-initials-eur-king-2-yob.txt > original-initials-eur-king-2-yob-check.txt  
       awk ‘$10==”PROBLEM” {print $0}’ original-initials-eur-king-2-yob-check.txt > original-initials-eur-king-2-yob-problem.txt
    2. Sex check  
       ./match.pl -f original-initials-eur-king-2-parents.fam -g original-initials-eur-king-2-parents.fam -k 2 -l 3 -v 5 > original-initials-eur-king-2-father-sex.txt  
       ./match.pl -f original-initials-eur-king-2-parents.fam -g original-initials-eur-king-2-father-sex.txt -k 2 -l 4 -v 5 > original-initials-eur-king-2-sex.txt  
       rm original-initials-eur-king-2-father-sex.txt  
       awk ‘{if ($7==2 || $8==1) print $0, “PROBLEM”; else print $0, “OK”}’ original-initials-eur-king-2-sex.txt > original-initials-eur-king-2-sex-check.txt  
       awk ‘$10==”PROBLEM” {print $0}’ original-initials-eur-king-2-sex-check.txt > original-initials-eur-king-2-sex-problem.txt
    3. Check the number of problematic families. Use the following commands to get the number:  
       wc -l original-initials-eur-king-2-yob-problem.txt  
       wc -l original-initials-eur-king-2-sex-problem.txt

### Examine the relationships within families (kin file)

* + 1. Identify any instances where the inferred relationships do not match those reported in MoBa:  
       awk ‘$16>0 {print $0}’ original-initials-eur-king-2.kin > original-initials-eur-king-2.kin-errors  
       ./match.pl -f /tsd/p697/data/durable/projects/moba\_qc\_imputation/resources/genotyped\_pedigree.txt -g original-initials-eur-king-2.kin-errors -k 4 -l 2 -v 8 > original-initials-eur-king-2.kin-errors-role1  
       ./match.pl -f /tsd/p697/data/durable/projects/moba\_qc\_imputation/resources/genotyped\_pedigree.txt -g original-initials-eur-king-2.kin-errors-role1 -k 4 -l 3 -v 8 > original-initials-eur-king-2.kin-errors  
       rm original-initials-eur-king-2.kin-errors-role1  
       wc -l original-initials-eur-king-2.kin-errors
       1. Identify any inferred duplicates/MZ twins that were unexpected:  
          awk ‘$15==”Dup/MZ” {print $0}’ original-initials-eur-king-2.kin-errors > original-initials-eur-king-2.kin-errors-MZ  
          wc -l original-initials-eur-king-2.kin-errors-MZ
       2. Identify any inferred parent-offspring relationships that were unexpected:  
          awk ‘$15==”PO” {print $0}’ original-initials-eur-king-2.kin-errors > original-initials-eur-king-2.kin-errors-PO  
          wc -l original-initials-eur-king-2.kin-errors-PO
       3. Identify any inferred full siblings that were unexpected:  
          awk ‘$15==”FS” {print $0}’ original-initials-eur-king-2.kin-errors > original-initials-eur-king-2.kin-errors-FS  
          wc -l original-initials-eur-king-2.kin-errors-FS
       4. Identify any inferred second degree relatives that were unexpected:  
          awk ‘$15==”2nd” {print $0}’ original-initials-eur-king-2.kin-errors > original-initials-eur-king-2.kin-errors-2nd  
          wc -l original-initials-eur-king-2.kin-errors-2nd
       5. Identify any inferred third degree relatives that were unexpected:  
          awk ‘$15==”3rd” {print $0}’ original-initials-eur-king-2.kin-errors > original-initials-eur-king-2.kin-errors-3rd  
          wc -l original-initials-eur-king-2.kin-errors-3rd
       6. Identify any inferred fourth degree relatives that were unexpected:  
          awk ‘$15==”4th” {print $0}’ original-initials-eur-king-2.kin-errors > original-initials-eur-king-2.kin-errors-4th  
          wc -l original-initials-eur-king-2.kin-errors-4th
       7. Identify any inferred unrelated individuals that were unexpected:  
          awk ‘$15==”UN” {print $0}’ original-initials-eur-king-2.kin-errors > original-initials-eur-king-2.kin-errors-UN  
          wc -l original-initials-eur-king-2.kin-errors-UN

### Examine the relationships between families (kin0 file)

* + 1. Identify any instances where the inferred relationships do not match those reported in MoBa:  
       awk ‘{if ($14==”Dup/MZ” || $14==”PO” || $14==”FS”) print $0}’ original-initials-eur-king-2.kin0 > original-initials-eur-king-2.kin0-errors  
       ./match.pl -f /tsd/p697/data/durable/projects/moba\_qc\_imputation/resources/genotyped\_pedigree.txt -g original-initials-eur-king-2.kin0-errors -k 4 -l 2 -v 8 > original-initials-eur-king-2.kin0-errors-role1  
       ./match.pl -f /tsd/p697/data/durable/projects/moba\_qc\_imputation/resources/genotyped\_pedigree.txt -g original-initials-eur-king-2.kin0-errors-role1 -k 4 -l 4 -v 8 > original-initials-eur-king-2.kin0-errors  
       rm original-initials-eur-king-2.kin0-errors-role1  
       wc -l original-initials-eur-king-2.kin0-errors
       1. Identify any inferred duplicates/MZ twins that were unexpected:  
          awk ‘$14==”Dup/MZ” {print $0}’ original-initials-eur-king-2.kin0-errors > original-initials-eur-king-2.kin0-errors-MZ  
          wc -l original-initials-eur-king-2.kin0-errors-MZ
       2. Identify any inferred parent-offspring relationships that were unexpected:  
          awk ‘$14==”PO” {print $0}’ original-initials-eur-king-2.kin0-errors > original-initials-eur-king-2.kin0-errors-PO  
          wc -l original-initials-eur-king-2.kin0-errors-PO
       3. Identify any inferred full siblings that were unexpected:  
          awk ‘$14==”FS” {print $0}’ original-initials-eur-king-2.kin0-errors > original-initials-eur-king-2.kin0-errors-FS  
          wc -l original-initials-eur-king-2.kin0-errors-FS

### Plot the relationships

* + 1. Make histograms of the estimated kinship coefficients between families
       1. Create file that is smaller for reading into R  
          awk ‘{print $2, $4, $19}’ original-initials-eur-king-2.ibs0 > original-initials-eur-king-2.ibs0\_hist
       2. Run plot-kinship-histogram.R according to the instructions in $GITHUB/lib/README-for-plot-kinship-histograms.txt. Please specify the following: input: original-initials-eur-king-2.ibs0\_hist and output: original-initials-eur-king-2-hist  
          Example: Rscript $GITHUB/lib/plot-kinship-histogram.R m12good-ec-eur-king-2.ibs0\_hist m12good-ec-eur-king-2-hist
       3. Remove the file used to plot the kinship histogram  
          rm original-initials-eur-king-2.ibs0\_hist
       4. Please post the graph on slack, copy it to the export folder for module II, and record the name of the plot in the google sheet.
    2. Make plots to visualize the between and within family relationships as reported by MoBa vs inferred by KING
    3. Please run the create-relplot.sh script that uses the relplot R script from KING, and generates a png file with four or two merged subplots with optionally customized legend positions. Usage information is on github (<https://github.com/norment/moba_qc_imputation/tree/master/tools#create-relplotsh>).  
       Usage: sh $GITHUB/tools/create-relplot.sh r\_relplot tag [legendpos1 legendpos2 legendpos3 legendpos4]  
       Arguments:  
       r\_relplot - R script file for relplot from KING  
       tag - a tag of data shown in the titles of the plots  
       legendpos1 - legend position of plot 1: topleft, topright, bottomleft, bottomright  
       legendpos2 - legend position of plot 2: topleft, topright, bottomleft, bottomright  
       legendpos3 - legend position of plot 3: topleft, topright, bottomleft, bottomright  
       legendpos4 - legend position of plot 4: topleft, topright, bottomleft, bottomright  
       Example: sh create-relplot.sh rotterdam1-yc-eur-king-2\_relplot.R "Rotterdam1 EUR, round 2" topright bottomright topright bottomright
    4. Post the plots on slack, copy the plot file to the export folder for plots of Module II, and record the plot name in the google sheet for report production.

### Fix within and between family issues

* + 1. Let Elizabeth know once you have completed the above steps and she will investigate the pedigree errors and create the update and remove files.
    2. Remove individuals with impossible relationships and update family and individual ids.  
       plink --bfile original-initials-eur-king-2-parents --remove original-initials-eur-king-2-unexpected-relationships.txt --update-ids original-initials-eur-king-2-fix-ids.txt --make-bed --out original-initials-eur-king-2-fix-ids
    3. Update paternal and maternal ids.  
       plink --bfile original-initials-eur-king-2-fix-ids --update-parents original-initials-eur-king-2-fix-parents.txt --make-bed --out original-initials-eur-king-2-fix-parents

### Identify any pedigree issues

* + 1. Create the covariate file
       1. Copy the age.txt file to your working directory.
       2. Update the FID to the match those updated during the above KING analysis.  
          ./match.pl -f original-initials-eur-king-2-fix-parents.fam -g age.txt -k 2 -l 2 -v 1 | awk ‘$4!=”-” {print $4, $2, $3}’ > original-initials-eur-king-2-fix-parents.cov  
          sed -i ‘1 i FID IID Age’ original-initials-eur-king-2-fix-parents.cov
    2. Run KING build using the following command:  
       /cluster/projects/p697/projects/moba\_qc\_imputation/software/king225 -b original-initials-eur-king-2-fix-parents.bed --build --degree 2 --prefix original-initials-eur-king-2.5 > original-initials-eur-king-2.5-slurm.txt
    3. Identify any issues
       1. Go through the build log file and see if there were any pedigree errors reported.
       2. See if an update ids or parents file was produced.
    4. Let Elizabeth know if there were any issues and she will investigate what is going on.
    5. Remove individuals with impossible relationships and update family and individual ids.
       1. Remove individuals with impossible relationships and update family and individual ids.  
          plink --bfile original-initials-eur-king-2-fix-parents --remove original-initials-eur-king-2.5-unexpected-relationships.txt --update-ids original-initials-eur-king-2.5-fix-ids.txt --make-bed --out original-initials-eur-king-2.5-fix-ids
       2. Update paternal and maternal ids.  
          plink --bfile original-initials-eur-king-2.5-fix-ids --update-parents original-initials-eur-king-2.5-fix-parents.txt --make-bed --out original-initials-eur-king-2.5-fix-parents
       3. If you had individuals removed, get the IDs of the removed individuals and add them to the IDs of individuals who were removed previously:  
          ./match.pl -f original-initials-eur-king-2.5-fix-parents.fam -g original-initials-eur-king-2-fix-parents.fam -k 2 -l 2 -v 1 | awk ‘$7==”-” {print $1,$2,”pedigree and known relatedness, round 2”}’ >> original-initials-eur-2-removed-individuals.txt

## Cryptic relatedness

In this section, we aim to remove individuals who share too much kinship with too many people.

* 1. Create files with (1) counts of individuals with whom each individual shares >=2.5% kinship and (2) sum of all kinship coefficients >=2.5% per individual
     1. Copy “cryptic.sh” file from “scripts” folder to your working directory and make the file executable:  
        chmod +x cryptic.sh
     2. Run the cryptic.sh script using the ibs0 output from KING.  
        ./cryptic.sh original-initials-eur-king-2.ibs0 original-initials-eur-cryptic-2
  2. Create plots
     1. Run the $GITHUB/lib/plot-cryptic.R script. See the “README-for-cryptic-plot” file for usage instructions.  
        For input1, please use “original-initials-eur-cryptic-2-kinship-sum.txt” file  
        For input2, please use “original-initials-eur-cryptic-2-counts.txt” file  
        For tag, please use the tag from plot-PLINK file  
        For output, please use original-initials-eur-cryptic-2  
        Example command: Rscript $GITHUB/lib/plot-cryptic.R m24-tz-eur-cryptic-2-kinship-sum.txt m24-tz-eur-cryptic-2-counts.txt "M24 EUR" m24-tz-eur-cryptic-2
     2. Please post your plot on slack, where we’ll determine what threshold would fit the data in your batch the best.
  3. Remove outliers
     1. Identify cryptic relatedness outliers. If your threshold is, for example, 15 for the sum of kinship, the example commands would be:  
        awk ‘$2>15 {print $1}’ original-initials-eur-cryptic-2-kinship-sum.txt > original-initials-eur-cryptic-2-sum-remove  
        ./match.pl -f original-initials-eur-king-2.5-fix-parents.fam -g original-initials-eur-cryptic-2-sum-remove -k 2 -l 1 -v 1 | awk ‘{print $2, $1}’ > original-initials-eur-cryptic-2-sum-remove.txt
     2. Remove cryptic relatedness outliers.  
        plink --bfile original-initials-eur-king-2.5-fix-parents --remove original-initials-eur-cryptic-2-sum-remove.txt --make-bed --out original-initials-eur-cryptic-clean-2
        1. Please make sure that the number of individuals removed (as shown in the log file) corresponds or is in line with the number of individuals in original-initials-eur-cryptic-2-sum-remove.txt file.  
           wc -l original-initials-eur-cryptic-2-sum-remove.txt
        2. Please record the number of individuals removed due to cryptic relatedness.
        3. If you had individuals removed, get the IDs of the removed individuals and add them to the IDs of individuals who were removed previously:  
           ./match.pl -f original-initials-eur-king-2.5-fix-parents.fam -g original-initials-eur-cryptic-clean-2.fam -k 2 -l 2 -v 1 | awk ‘$7==”-” {print $1,$2,”cryptic relatedness, round 2”}’ >> original-initials-eur-2-removed-individuals.txt

## Mendelian errors

* 1. Remove families with more than 5% Mendel errors and SNPs with more than 1% of Mendel errors and zeros out the other minor Mendel errors.
     1. If your fam file contains no individuals with unknown sex run the below command then move to step 12.2  
        plink --bfile original-initials-eur-cryptic-clean-2 --me 0.05 0.01 --set-me-missing --mendel-duos --make-bed --out original-initials-eur-me-clean-sex-2
     2. If your fam file contains individuals with unknown sex, use the PLINK inferred sex for people whose sex is missing.  
        awk ‘$3==0 {print $1,$2,$4}’ original-initials-eur-2-sexcheck.sexcheck > original-initials-eur-2-sex-me.txt  
        plink --bfile original-initials-eur-cryptic-clean-2 --update-sex original-initials-eur-2-sex-me.txt --me 0.05 0.01 --set-me-missing --mendel-duos --make-bed --out original-initials-eur-me-clean-2
     3. Restore the sex that was before update for ME check purposes:  
        awk ‘{print $1,$2,0}’ original-initials-eur-2-sex-me.txt > original-initials-eur-2-sex-me-back.txt
     4. plink --bfile original-initials-eur-me-clean-2 --update-sex original-initials-eur-2-sex-me-back.txt --make-bed --out original-initials-eur-me-clean-sex-2
  2. Record the number of families removed, the number of SNPs removed and the number of Mendelian errors zero-ed out (all these numbers can be obtained from the plink log file of step 12.1).
  3. If you had individuals (families) removed, get the IDs of the removed individuals and add them to the IDs of individuals who were removed previously:  
     ./match.pl -f original-initials-eur-me-clean-sex-2.fam -g original-initials-eur-cryptic-clean-2.fam -k 2 -l 2 -v 1 | awk ‘$7==”-” {print $1,$2,”Mendelian-error, round 2”}’ >> original-initials-eur-2-removed-individuals.txt
  4. If you had SNPs removed, add their IDs to those that were removed previously:  
     ./match.pl -f original-initials-eur-me-clean-sex-2.bim -g original-initials-eur-cryptic-clean-2.bim -k 2 -l 2 -v 1 | awk ‘$7==”-” {print $2,”Mendelian-error, round 2”}’ >> original-initials-eur-2-bad-snps.txt

## PCA with 1KG

### Prune the data and remove long stretches of LD

* + 1. Prune the data  
       plink --bfile original-initials-eur-me-clean-sex-2 --indep-pairwise 3000 1500 0.1 --out original-initials-eur-me-clean-sex-2-indep
       1. If needed, repeat the pruning to remove residual LD, until there are about 100,000 SNPs left.
    2. Extract the set of pruned SNPs  
       plink --bfile original-initials-eur-me-clean-sex-2 --extract original-initials-eur-me-clean-sex-2-indep.prune.in --make-bed --out original-initials-eur-me-clean-sex-2-pruned
    3. Long LD regions
       1. Copy the file containing the list of long LD regions (Build37) from the “resources” folder to your working directory (name of the file is “high-ld.txt”).
       2. Create high LD set  
          plink --bfile original-initials-eur-me-clean-sex-2-pruned --make-set high-ld.txt --write-set --out original-initials-eur-me-clean-sex-2-highld
       3. Exclude SNPs in high LD set  
          plink --bfile original-initials-eur-me-clean-sex-2-pruned --exclude original-initials-eur-me-clean-sex-2-highld.set --make-bed --out original-initials-eur-me-clean-sex-2-trimmed

### Identify SNPs overlapping with 1KG

* + 1. Copy the 1KG PLINK bfiles from the “resources” folder in DATA/DURABLE to your working directory (1kg.bed,1kg.bim and 1kg.fam); these files have the strand ambiguous SNPs already removed.
    2. Run the following commands to identify the overlapping SNPs  
       cut -f2 1kg.bim | sort -s > 1kg.bim.sorted  
       cut -f2 original-initials-eur-me-clean-sex-2-trimmed.bim | sort -s > original-initials-eur-me-clean-sex-2-trimmed.bim.sorted  
       join 1kg.bim.sorted original-initials-eur-me-clean-sex-2-trimmed.bim.sorted > original-initials-eur-me-clean-sex-2-1kg-snps.txt  
       rm original-initials-eur-me-clean-sex-2-trimmed.bim.sorted
    3. Record the number of SNPs you have common in your batch and in 1KG:  
       wc -l original-initials-eur-me-clean-sex-2-1kg-snps.txt

### Merge with the 1KG dataset

* + 1. Extract the overlapping SNPs
       1. In your batch:  
          plink --bfile original-initials-eur-me-clean-sex-2-trimmed --extract original-initials-eur-me-clean-sex-2-1kg-snps.txt --make-bed --out original-initials-eur-me-clean-sex-2-1kg-common
       2. In the 1KG dataset:  
          plink --bfile 1kg --extract original-initials-eur-me-clean-sex-2-1kg-snps.txt --make-bed --out 1kg-original-initials-eur-me-clean-sex-2-common
    2. Merge the bfiles  
       plink --bfile original-initials-eur-me-clean-sex-2-1kg-common --bmerge 1kg-original-initials-eur-me-clean-sex-2-common --make-bed --out original-initials-eur-me-clean-sex-2-1kg-merged
       1. If you have SNPs that have 3+ alleles, flip those alleles in 1kg data and merge again. To flip, run the following command:  
          plink --bfile 1kg-original-initials-eur-me-clean-sex-2-common --flip original-initials-eur-me-clean-sex-2-1kg-merged-merge.missnp --make-bed --out 1kg-original-initials-eur-me-clean-sex-2-common-flip  
          To merge again, run the following command:  
          plink --bfile original-initials-eur-me-clean-sex-2-1kg-common --bmerge 1kg-original-initials-eur-me-clean-sex-2-common-flip --make-bed --out original-initials-eur-me-clean-sex-2-1kg-second-merged
       2. If you still have SNPs with 3+ alleles after merging with flipped data, remove those SNPs from both 1KG and MoBa data. To remove the SNPs with 3+ alleles from MoBa and from 1kg, run the following commands:  
          plink --bfile original-initials-eur-me-clean-sex-2-1kg-common --exclude original-initials-eur-me-clean-sex-2-1kg-second-merged-merge.missnp --make-bed --out original-initials-eur-me-clean-sex-2-1kg-common-clean  
          plink --bfile 1kg-original-initials-eur-me-clean-sex-2-common-flip --exclude original-initials-eur-me-clean-sex-2-1kg-second-merged-merge.missnp --make-bed --out 1kg-original-initials-eur-me-clean-sex-2-common-flip-clean  
          To merge again, run the following command:  
          plink --bfile original-initials-eur-me-clean-sex-2-1kg-common-clean --bmerge 1kg-original-initials-eur-me-clean-sex-2-common-flip-clean --make-bed --out original-initials-eur-me-clean-sex-2-1kg-clean-merged
    3. Record how many SNPs you are going to use for PCA:  
       wc -l original-initials-eur-me-clean-sex-2-1kg-clean-merged.bim

### PCA

* + 1. Copy the populations.txt file from “resources” folder on DATA/DURABLE to your working directory. populations.txt is a text file containing the population, based on which PLINK will calculate the main PCs, in this case it will be just one word “parent”).
    2. Create the original-initials-eur-me-clean-sex-2-fam-populations.txt – a text file with 3 columns: family ID, individual ID and so-called population, in this case it will be “parent” for unrelated individuals and “child” for related individuals. To create this file, do the following:  
       awk ‘{if($3==0 && $4==0) print $1,$2,”parent”; else print $1,$2,”child”}’ original-initials-eur-me-clean-sex-2-1kg-merged.fam > original-initials-eur-me-clean-sex-2-fam-populations.txt
       1. If you did the flip, use original-initials-eur-me-clean-sex-2-1kg-second-merged.fam
       2. If you had to remove SNPs after the flip, use original-initials-eur-me-clean-sex-2-1kg-clean-merged.fam
    3. Run the PCA  
       plink --bfile original-initials-eur-me-clean-sex-2-1kg-merged --pca --within original-initials-eur-me-clean-sex-2-fam-populations.txt --pca-clusters populations.txt --out original-initials-eur-me-clean-sex-2-1kg-pca
       1. If your data had SNPs with 3+ alleles and you did flip+merge steps, use the original-initials-eur-me-clean-sex-2-1kg-second-merged bfiles
       2. If your data had SNPs with 3+ alleles after flip+merge steps and you did remove+merge steps, then use the original-initials-eur-me-clean-sex-2-1kg-clean-merged bfiles
       3. If your batch is large and you need to run the PCA on Colossus, copy your PLINK input files into Colossus and use the PCA\_POP.job

### Plot PCs

* + 1. The individuals in 1kg.fam file are not ordered according to their population. But in order to assign colors during plotting, we need them to be in order. Thus, we will divide the file containing the PCs into “original-eur” and “1kg” portions, sort the “1kg” portion and combine the two together for plotting. This can be achieved with the following commands:  
       sort -k2 original-initials-eur-me-clean-sex-2-1kg-pca.eigenvec > original-initials-eur-me-clean-sex-2-1kg-pca-sorted  
       head -n “a” original-initials-eur-me-clean-sex-2-1kg-pca-sorted > original-initials-eur-me-clean-sex-2-1kg-pca-1  
       NB the “a” in the “head” command is the number of individuals in the “original” PLINK files that were merged with 1KG (it is the number of individuals in original-initials-eur-me-clean-sex-2-trimmed.fam file)  
       tail -n 1083 original-initials-eur-me-clean-sex-2-1kg-pca-sorted | sort -k2 > 1kg-initials-eur-2-pca-1  
       NB 1083 is the number of individuals in 1kg.fam  
       cat original-initials-eur-me-clean-sex-2-1kg-pca-1 1kg-initials-eur-2-pca-1 > original-initials-eur-me-clean-sex-2-1kg-pca
    2. Plot your batch with the 1KG dataset.  
       The code for plotting is located in $GITHUB/lib/plot-pca-with-1kg.R. Instructions are in $GITHUB/lib/README.md
       1. Please remember to use the “plot-PLINK” file in “resources” folder to assign the harmonized tag to the plots of your batch.  
          Please specify the “outprefix” as “original-initials-eur-me-clean-sex-2-1kg”.
       2. When you’ve made the plots, please post them on slack and place a copy in the export folder for Module II.
    3. If needed, select a cleaner subsample using the script named “$GITHUB/lib/select-subsamples-on-pca.R”. Please name the customfile as “original-initials-eur-pca-core-select-3-custom.txt” and the “outprefix” as “original-initials-eur-selection-3”.  
       For examples of <customfile> see in $GITHUB/config folder.
       1. Please note that here we select only EUR subsample (in other words, we tighten the previous selection). Thus, the customfile will have only EUR section. Your customfile may look like this:  
          eur\_zoom\_threshold: PC1> -0.01  
          eur\_draw\_threshold: PC1>0  
          eur\_legend\_position: bottomleft
       2. In this example, the selection is done based on PC1 only. Please add PC2 and/or other PCs as required.
    4. Once done, please copy the plots to /tsd/p697/data/durable/projects/moba\_qc\_imputation/export/Module\_II\_Plots folder.
    5. Please record the names of the plots in MoBa\_QC\_numbers sheet.

### Remove ancestry outliers

* + 1. Use the following command to remove PC outliers:  
       plink --bfile original-initials-eur-me-clean-sex-2 --keep original-initials-eur-selection-3-core-subsample-eur.txt --make-bed --out original-initials-eur-3-keep
    2. Please record the number of individuals kept in MoBa\_QC\_numbers sheet (you can see the number in the log file of the above PLINK command).
    3. Add the IDs of individuals removed to those who were removed previously:  
       ./match.pl -f original-initials-eur-3-keep.fam -g original-initials-eur-me-clean-sex-2.fam -k 2 -l 2 -v 1 | awk ‘$7==”-” {print $1,$2,”PCA-with-1kg-round-2”}’>> original-initials-eur-2-removed-individuals.txt
    4. If there are no outliers in this step, then take the original-initials-eur-me-clean-sex-2 biles into step 14.

## PCA without 1KG

### Prune and remove long stretches of LD

* + 1. Prune the data
       1. If there were outliers:  
          plink --bfile original-initials-eur-3-keep --indep-pairwise 3000 1500 0.1 --out original-initials-eur-3-keep-indep
       2. If there were no outliers in step 13, then run the following command:  
          plink --bfile original-initials-eur-me-clean-sex-2 --indep-pairwise 3000 1500 0.1 --out original-initials-eur-3-keep-indep
    2. Check the number of remaining SNPs after pruning:  
       wc -l original-initials-eur-3-keep-indep.prune.in
       1. If the number is substantially larger than 100K, repeat the pruning until there are about 100K SNPs left.
    3. Remove long stretches of LD  
       plink --bfile original-initials-eur-3-keep --extract original-initials-eur-3-keep-indep.prune.in --make-set high-ld.txt --write-set --out original-initials-eur-3-keep-highld  
       plink --bfile original-initials-eur-3-keep --extract original-initials-eur-3-keep-indep.prune.in --exclude original-initials-eur-3-keep-highld.set --make-bed --out original-initials-eur-3-trimmed
    4. Record the number of SNPs in original-initials-eur-3-trimmed.bim  
       wc -l original-initials-eur-3-trimmed.bim

### PCA

* + 1. Copy the populations.txt file from “resources” folder on DATA/DURABLE to your working directory. populations.txt is a text file containing the population, based on which PLINK will calculate the main PCs, in this case it will be just one word “parent”).
    2. Create original-initials-eur-3-populations.txt – a text file with 3 columns: family ID, individual ID and so-called population, in this case it will be “parent” for unrelated individuals and “child” for related individuals.  
       awk ‘{if($3==0 && $4==0) print $1,$2,”parent”; else print $1,$2,”child”}’ original-initials-eur-3-trimmed.fam > original-initials-eur-3-populations.txt
    3. Run PCA in PLINK  
       plink --bfile original-initials-eur-3-trimmed --pca --within original-initials-eur-3-populations.txt --pca-clusters populations.txt --out original-initials-eur-3-pca
       1. If your batch is large and you need to run the PCA on Colossus, copy your PLINK input files into Colossus and use the PCA\_POP.job

### Plot PCs

* + 1. Identify founders and non-founders  
       awk ‘{if($3==0 && $4==0) print $1,$2,”black”; else print $1,$2,”red” }’ original-initials-eur-3-keep.fam > original-initials-eur-3-keep-fam.txt  
       ./match.pl -f original-initials-eur-3-keep-fam.txt -g original-initials-eur-3-pca.eigenvec -k 2 -l 2 -v 3 > original-initials-eur-3-keep-pca-fam.txt
    2. Create PC plots using the plot-batch-PCs.R script. See $GITHUB/lib/README-for-plot-batch-PCs for usage instructions. The input is original-initials-eur-3-keep-pca-fam.txt, title is “original EUR, round 2”, and output is original-initials-eur-3-pca.png.
    3. Once done, please post the plots on slack, and copy the plots to the /tsd/p697/data/durable/projects/moba\_qc\_imputation/export/Module\_II\_Plots folder.
    4. Please record the names of the plots in MoBa\_QC\_numbers sheet.

### Remove PC outliers

* + 1. Select the individuals to keep (in this example, we select individuals with PC1 value more than zero; please adjust the awk command as needed). Please reach out to Elizabeth if you have any doubts/questions.  
       awk ‘$3>0 {print $1,$2}’ original-initials-eur-3-pca.eigenvec > original-initials-eur-3-pca-keep.txt
    2. Create PC plots after outlier selection
       1. Create a file containing the individuals kept with founders and non-founder status, using the same threshold conditions as step 14.4.1.  
          awk ‘$3>0 {print $0}’ original-initials-eur-3-keep-pca-fam.txt > original-initials-eur-3-pca-fam-keep.txt
       2. Run the plot-batch-PCs.R script. See $GITHUB/lib/README-for-plot-batch-PCs for usage instructions. The input is original-initials-eur-3-pca-fam-keep.txt, title is “original EUR, round 2”, and output is original-initials-eur-3-pca-keep.png
       3. Once done, please post the plots on slack, and copy the plots to the /tsd/p697/data/durable/projects/moba\_qc\_imputation/export/Module\_II\_Plots folder.
       4. Please record the names of the plots in MoBa\_QC\_numbers sheet.
    3. Create PLINK files without the outliers  
       plink --bfile original-initials-eur-3-keep --keep original-initials-eur-3-pca-keep.txt --make-bed --out original-initials-eur-2-round-selection
    4. Please record the number of individuals kept in MoBa\_QC\_numbers sheet (you can see the number in the log file of the above PLINK command).
    5. If you had outliers, add their IDs to those who were removed previously:  
       ./match.pl -f original-initials-eur-2-round-selection.fam -g original-initials-eur-3-keep.fam -k 2 -l 2 -v 1 | awk ‘$7==”-” {print $1,$2,”PCA-without-1kg-round-2”}’>> original-initials-eur-2-removed-individuals.txt

## Plate effects

If you had outliers in step 14, use the original-initials-eur-2-round-selection bfiles. If you did not have any outliers use the original-initials-eur-3-keep bfiles.

### PCA

* + 1. Prune the data  
       plink --bfile original-initials-eur-2-round-selection --indep-pairwise 3000 1500 0.1 --out original-initials-eur-2-round-indep
    2. Remove long stretches of LD and extract the SNPs
       1. Copy the file containing the list of long LD regions (Build37) from the “resources” folder to your working directory (name of the file is “high-ld.txt”)
       2. Extract pruned SNPs and Remove long stretches of LD  
          plink --bfile original-initials-eur-2-round-selection --extract original-initials-eur-2-round-indep.prune.in --make-set high-ld.txt --write-set --out original-initials-eur-2-round-highld  
          plink --bfile original-initials-eur-2-round-selection --extract original-initials-eur-2-round-indep.prune.in --exclude original-initials-eur-2-round-highld.set --make-bed --out original-initials-eur-2-round-trimmed
    3. Copy the populations.txt file from “resources” folder on DATA/DURABLE to your working directory. populations.txt is a text file containing the population, based on which PLINK will calculate the main PCs, in this case it will be just one word “parent”).
    4. Run the PCA  
       plink --bfile original-initials-eur-2-round-trimmed --pca --within original-initials-eur-3-populations.txt --pca-clusters populations.txt --out original-initials-eur-2-round-pca

### Plot PCs by plate

* + 1. Copy the plate file relevant to your array from “resources” folder to your working directory: (HCE-plates.txt, GSA-plates.txt, or OMNI-plates.txt). These files contain the information on where each individual’s DNA was plated. Throughout this section the HCE-plates.txt file is used as an example. Please make sure to use update the code to match the array used to genotype your batch.
    2. Add plate information to a file with PCA results:  
       ./match.pl -f HCE-plates.txt -g original-initials-eur-2-round-pca.eigenvec -k 1 -l 2 -v 3 | awk ‘$23!=”-” {print $0}’ | sort -k23 > original-initials-eur-3-pca-plates.txt  
       awk ‘{print $1,$2,$23}’ original-initials-eur-3-pca-plates.txt > original-initials-eur-3-plate-groups.txt
    3. Create exploratory plots of PCs colored by plate using the plot-PC-by-plate.R script. See $GITHUB/lib/README-for-plot-PC-by-plate for usage instructions. The input is original-initials-eur-3-pca-plates.txt, the title is “original EUR, round 2”, and the output is original-initials-eur-3  
       Example: Rscript $GITHUB/lib/plot-PC-by-plate.R m24-tz-eur-3-pca-plates.txt “m24 EUR, round 2” m24-tz-eur
    4. Once done, please post the plots on slack, and copy the plots to the /tsd/p697/data/durable/projects/moba\_qc\_imputation/export/Module\_II\_Plots folder.
    5. Please record the names of the plots in the MoBa\_QC\_numbers sheet.

### ANOVA

* + 1. To perform ANOVA for the first 10PCs and the plates run the anova-for-PC-vs-plates.R script. See $GITHUB/lib/README-for-anova-for-PC-vs-plates for usage instructions. Please specify the input as original-initials-eur-3-pca-plates.txt and the output as original-initials-eur-3-pca-anova-results.txt
    2. Once done, please post the results on slack. You may use the following command to see the results:  
       more original-initials-eur-3-pca-anova-results.txt
    3. Please record the p-values in the MoBa\_QC\_numbers sheet.

### Test for association between the plate and SNPs

* + 1. Run a Cochran-Mantel-Haenszel test for association in founders using sex as the phenotype.  
       plink --bfile original-initials-eur-2-round-selection --filter-founders --chr 1-22 --pheno original-initials-eur-2-round-selection.fam --mpheno 3 --within original-initials-eur-3-plate-groups.txt --mh2 --out original-initials-eur-3-mh-plates
    2. Create QQ plot using the $GITHUB/lib/plot-qqplot.R script for more information see: <https://github.com/norment/moba_qc_imputation/tree/master/lib#plot-qqplotr>. Usage: Rscript plot-qqplot.R inputfile tag pcol outprefix.  
       Please use the following arguments: inputfile: original-initials-eur-3-mh-plates.cmh2; tag: use the “plot-PLINK” file in “resources” folder to assign the harmonized tag to the plots of your batch, followed by the core subpopulation, and QC round; outprefix: original-initials-eur-3-mh-plates-qq-plot.  
       Example: Rscript $GITHUB/lib/plot-qqplot.R original-initials-eur-3-mh-plates.cmh2 “PLOT-PLINK tag, population, round 2” 5, original-initials-eur-3-mh-plates-qq-plot
    3. Once done, please post the plots on slack, and copy the plots to the /tsd/p697/data/durable/projects/moba\_qc\_imputation/export/Module\_II\_Plots folder.
    4. Please record the name of the plot in the MoBa\_QC\_numbers sheet.
    5. Check if there are any significant SNPs at p-value <0.001 threshold  
       sort -k5 -g original-initials-eur-3-mh-plates.cmh2 | grep -v “NA” > original-initials-eur-3-mh2-plates-sorted  
       awk ‘$5<0.001 {print $2}’ original-initials-eur-3-mh2-plates-sorted > original-initials-eur-3-mh2-plates-significant
    6. Record how many SNPs are significant (with p<0.001) and post that number on slack:  
       wc -l original-initials-eur-3-mh2-plates-significant
    7. Remove the SNPs with significant difference between plates
       1. If you had no outliers in step 14:  
          plink --bfile original-initials-eur-3-keep --exclude original-initials-eur-3-mh2-plates-significant --make-bed --out original-initials-eur-3-batch
       2. If you had outliers in step 14:  
          plink --bfile original-initials-eur-2-round-selection --exclude original-initials-eur-3-mh2-plates-significant --make-bed --out original-initials-eur-3-batch
    8. Add the IDs of SNPs removed here to those that were removed previously:  
       awk ‘{print $1,”plate-effect, round 2”}’ original-initials-eur-3-mh2-plates-significant >> original-initials-eur-2-bad-snps.txt

### Re-run PCA

* + 1. Prune the data  
       plink --bfile original-initials-eur-3-batch --indep-pairwise 3000 1500 0.1 --out original-initials-eur-3-batch-indep
    2. Remove long stretches of LD and extract the SNPs  
       plink --bfile original-initials-eur-3-batch --extract original-initials-eur-3-batch-indep.prune.in --make-set high-ld.txt --write-set --out original-initials-eur-3-batch-highld  
       plink --bfile original-initials-eur-3-batch --extract original-initials-eur-3-batch-indep.prune.in --exclude original-initials-eur-3-batch-highld.set --make-bed --out original-initials-eur-3-batch-trimmed
    3. Copy the populations.txt file from “resources” folder on DATA/DURABLE to your working directory. populations.txt is a text file containing the population, based on which PLINK will calculate the main PCs, in this case it will be just one word “parent”).
    4. Run the PCA  
       plink --bfile original-initials-eur-3-batch-trimmed --pca --within original-initials-eur-3-populations.txt --pca-clusters populations.txt --out original-initials-eur-3-batch-pca

### Re-run ANOVA

* + 1. Add plate information to a file with PCA results:  
       ./match.pl -f HCE-plates.txt -g original-initials-eur-3-batch-pca.eigenvec -k 1 -l 2 -v 3 | awk ‘$23!=”-” {print $0}’ | sort -k23 > original-initials-eur-3-batch-pca-plates.txt
    2. To perform ANOVA for the first 10PCs and the plates run the anova-for-PC-vs-plates.R script. See $GITHUB/lib/README-for-anova-for-PC-vs-plates for usage instructions. Please specify the input as original-initials-eur-3-batch-pca-plates.txt and the output as original-initials-eur-3-batch-pca-anova-results.txt
    3. Once done, please post the results on slack. You may use the following command to see the results:  
       more original-initials-eur-3-batch-pca-anova-results.txt
    4. If there are no significant differences, proceed to round three.If you have significant differences, please let Elizabeth know.

# THIRD ROUND

Please let Elizabeth know on slack when you are entering round two of the QC.

## Basic QC

### Remove sex chromosomes

* + 1. Run the following command to remove variants with MAF <0.5% and sex chromosomes:  
       plink --bfile original-initials-eur-3-batch --chr 1-22 --make-bed --out original-initials-eur-round-3
    2. Record the number of SNPs removed.
    3. Make a list of IDs of SNPs removed:   
       ./match.pl -f original-initials-eur-round-3.bim -g original-initials-eur-3-batch.bim -k 2 -l 2 -v 1 | awk ‘$7==”-” {print $2,”sex chromosome, round 3”}’ > original-initials-eur-3-bad-snps.txt

### Remove rare variants (MAF<0.5%)

* + 1. Run the following command to remove variants with MAF <0.5%:  
       plink --bfile original-initials-eur-round-3 --chr 1-22 --maf 0.005 --make-bed --out original-initials-eur-3-common
    2. Record the number of SNPs removed.
    3. Make a list of IDs of SNPs removed:   
       ./match.pl -f original-initials-eur-3-common.bim -g original-initials-eur-round-3.bim -k 2 -l 2 -v 1 | awk ‘$7==”-” {print $2,”rare, round 3”}’ >> original-initials-eur-3-bad-snps.txt

### Call rates

* + 1. Histograms of missing rates
       1. Run the following command to identify the missing rates for SNPs and for individuals:  
          plink --bfile original-initials-eur-3-common --missing --out original-initials-eur-3-missing  
          This command will generate two files: one with extension of “.imiss” and another with extension of “.lmiss”. The “.imiss” file contains the missing rates per individual and the “.lmiss” file contains missing rates per SNP.
       2. Plot the missing rates in R using the script named “$GITHUB/lib/plot-missingness-histogram.R” (see the $GITHUB/lib/README.md file for more information about the script functionality and it’s input and output arguments). Run the script according to its manual.
          1. Example of how to run the plot-missingness-histogram.R script in your working directory (in terminal):  
             Rscript $GITHUB/lib/plot-missingness-histogram.R dataprefix “tag”  
             Where: dataprefix is the prefix of your .imiss and .lmiss files created in this step (specific to your batch and core subsample, i.e. original-initials-eur-missing).  
             tag (make sure it is written in quotation marks) consists of three arguments, first the name of your batch, second the name of the core subpopulation, and third the QC round.  
             Example of the command:  
             Rscript $GITHUB/lib/plot-missingness-histogram.R original-initials-eur-3-missing “tag population, round 3”
       3. Please record the names of the plots you’ve created in MoBa\_QC\_numbers spreadsheet on google drive and copy the files with plots to the export folder for module II.
    2. Remove SNPs with call rate <95%  
       plink --bfile original-initials-eur-3-common --geno 0.05 --make-bed --out original-initials-eur-3-95
    3. Record the number of failed SNPs.
    4. Make a list with IDs of the SNPs failed at this step:  
       ./match.pl -f original-initials-eur-3-95.bim -g original-initials-eur-3-common.bim -k 2 -l 2 -v 1 | awk ‘$7==”-” {print $2,”call-rate-below-95, round 3”}’ >> original-initials-eur-3-bad-snps.txt
    5. Remove SNPs with call rate below 98%:  
       plink --bfile original-initials-eur-3-95 --geno 0.02 --make-bed --out original-initials-eur-3-98
    6. Remove SNPs and individuals whose call rate is below 98%  
       plink --bfile original-initials-eur-3-98 --geno 0.02 --mind 0.02 --make-bed --out original-initials-eur-3-call-rates
    7. Please record the number of SNPs and individuals failing the call rates in the MoBa\_QC\_numbers sheet corresponding to your batch.
    8. Please make a file with IDs of the SNPs that failed the call rate in this step:  
       ./match.pl -f original-initials-eur-3-call-rates.bim -g original-initials-eur-3-95.bim -k 2 -l 2 -v 1| awk ‘$7==”-” {print $2,”call-rate-below-98, round 3”}’ >> original-initials-eur-3-bad-snps.txt
    9. Make a file with IDs of individuals to remove:   
       awk ‘{print $1,$2,”call-rate-below-98, round 3”}’ original-initials-eur-3-call-rates.irem > original-initials-eur-3-removed-individuals.txt

### HWE test

* + 1. HWE test, run the following command to remove SNPs not in HWE (with p<1.00E-06):  
       plink --bfile original-initials-eur-3-call-rates --hwe 0.000001 --make-bed --out original-initials-eur-3-basic-qc
    2. Please record the number of SNPs that fail the HWE filter
    3. Create a file with the IDs of SNPs that failed.  
       ./match.pl -f original-initials-eur-3-basic-qc.bim -g original-initials-eur-3-call-rates.bim -k 2 -l 2 -v 1| awk ‘$7==”-” {print $2,”out-of-HWE, round 3”}’ >> original-initials-eur-3-bad-snps.txt

### Heterozygosity

* + 1. Estimate heterozygosity and missingness with the following PLINK command:  
       plink --bfile original-initials-eur-3-basic-qc --het --missing --out original-initials-eur-3-common-het-miss
    2. Plot the data and make a list of outliers based on being outside the +/- 3 standard deviations of the sample mean.
       1. Use the script “$GITHUB/lib/plot-heterozygosity-common.R” to run in terminal. (see the $GITHUB/lib/README.md file for more information about the script functionality and it’s input and output arguments).
          1. Usage: Rscript $GITHUB/lib/plot-heterozygosity-common.R dataprefix “tag”  
             Where: dataprefix - prefix of the outputs from PLINK command to estimate heterozygosity and missingness  
             “tag” - “tag population, round 3”, please remember to use the tag from “plot-PLINK” file in “resources” folder.
          2. Please record the number of individuals who are the outliers, you can find the number by using the following command:  
             tail -n +2 original-initials-eur-3-common-het-miss-het-fail.txt | wc -l
    3. Remove heterozygosity outliers:  
       plink --bfile original-initials-eur-3-basic-qc --remove original-initials-eur-3-common-het-miss-het-fail.txt --make-bed --out original-initials-eur-3-het
    4. Add the IDs of outliers to those who were removed previously:  
       awk ‘{print $1,$2,”heterozygosity, round 3”}’ original-initials-eur-3-common-het-miss-het-fail.txt >> original-initials-eur-3-removed-individuals.txt

## Pedigree build and known relatedness

This can be a memory intensive step particularly if you have a large batch, therefore, it is recommended to run "king" directly on the p697-appn-norment01 machine or on Colossus. The software directory has multiple versions of "king", use the latest version: king225. NB king225rhel6 can be used on TSD login nodes (but only if submit nodes and app node are unavailable); king225rhel6 binary will crash on p697-submit nodes and p697-appn-norment01. Finally, king225\_patch1 is a customly modified "king" program which produces a smaller output file in "king --ibs". Only individuals with kinship above 0.025 are reported.

### KING pedigree build and relatedness

* + 1. Create the covariate file.
       1. Copy the age.txt file from “resources” folder to your working directory on DATA/DURABLE.
       2. Update the file to contain the FID from previous pedigree builds, and rename the file to the same prefix as the PLINK files you’ve copied to Colossus. For example, if the PLINK files you are using have prefix of original-initials-eur-3-het, then rename the file to original-initials-eur-3-het.cov. The command lines for this are:  
          ./match.pl -f original-initials-eur-3-het.fam -g age.txt -k 2 -l 2 -v 1 | awk ‘$4!=”-” {print $4, $2, $3}’ > original-initials-eur-3-het.cov  
          sed -i ‘1 i\FID IID Age’ original-initials-eur-3-het.cov
    2. If you are running KING on the p697-appn-norment01 machine use the following command:  
       /cluster/projects/p697/projects/moba\_qc\_imputation/software/king225 -b original-initials-eur-3-het.bed --related --ibs --build --degree 2 --rplot --prefix original-initials-eur-king-3 > original-initials-eur-king-3-slurm.txt
    3. If you are running KING on Colossus:
       1. Create a folder named with your initials on /cluster/projects/p697/projects/moba\_qc\_imputation (please see “folder-structure-moba-2020.pdf” for reference). It will be your working directory on Colossus. Please use it every time you run things on Colossus.
       2. Copy original-initials-eur-3-het.bed, original-initials-eur-3-het.bim and original-initials-eur-3-het.fam, and original-initials-eur-3-het.cov files to your working directory on Colossus.
       3. Once in your working directory on Colossus, define “fin” and “fout” environmental variables as shown below, adjusting the names to reflect the names of your files (e.g. “original-initials” needs to become “moba12good-ec”), and then submit the job to Colossus:  
          export fin=original-initials-eur-3-het  
          export fout=original-initials-eur-king-3  
          sbatch $GITHUB/jobs/KING.job
       4. When you submit the job, please note the job number. Once the job is finished, rename the slurm output file using the following command:  
          mv slurm-your.jobnumber.out original-initials-eur-king-3-slurm.txt
       5. Move KING output files to your working directory on DATA/DURABLE and delete the input files used for this step from your working directory on Colossus. These can be achieved with the following commands:  
          rm original-initials-eur-3-het.\*  
          mv original-initials-eur-king-3\*.\* /tsd/p697/data/durable/projects/moba\_qc\_imputation/initials
       6. Go back to your working directory on DATA/DURABLE and continue working in that directory for the rest of this step.
    4. Please check in the slurm file that age was taken into account when running the build section of KING. Please let Elizabeth know if age was not taken into account.

### Update pedigree according to KING output:

* + 1. Check that the IID, PID, and MID in the original-initials-eur-king-3updateids.txt or original-initials-eur-king-3updateparents.txt files have not been changed from SENTRIX format. If the IIDs have been changed from the SENTRIX format use the following commands to convert back to SENTRIX format. Otherwise proceed to step 17.2.2.
       1. Command to convert the updateids file:  
          awk '{print $1,$2,$3,$2}' original-initials-eur-king-3updateids.txt > original-initials-eur-king-3updateids.txt-sentrix
       2. Command to convert the updateparents file:  
          R  
          library(tidyr)  
          update <- read.table('original-initials-eur-king-3updateparents.txt',h=F)  
          update$Order <- 1:nrow(update)  
          update\_IID <- update[grep("->", update$V2),]  
          no\_update\_IID <- update[!update$Order %in% update\_IID$Order,]  
          update\_IID <- separate(update\_IID, V2, into=c(NA,"V2"), sep="->", remove=T)  
          update\_IID <- update\_IID[,c(1,3:9)]  
          update <- rbind(update\_IID, no\_update\_IID)  
          rm(update\_IID, no\_update\_IID)  
          update\_PID <- update[grep("->", update$V3),]  
          no\_update\_PID <- update[!update$Order %in% update\_PID$Order,]  
          update\_PID <- separate(update\_PID, V3, into=c(NA,"V3"), sep="->", remove=T)  
          update\_PID <- update\_PID[,c(1:2,4:9)]  
          update <- rbind(update\_PID, no\_update\_PID)  
          rm(update\_PID, no\_update\_PID)  
          update\_MID <- update[grep("->", update$V4),]  
          no\_update\_MID <- update[!update$Order %in% update\_MID$Order,]  
          update\_MID <- separate(update\_MID, V4, into=c(NA,"V4"), sep="->", remove=T)  
          update\_MID <- update\_MID[,c(1:3,5:9)]  
          update <- rbind(update\_MID, no\_update\_MID)  
          rm(update\_MID, no\_update\_MID)  
          update <- update[,c(1:7)]  
          write.table(update, 'original-initials-eur-king-3updateparents.txt-sentrix', row.names=F, col.names=F, sep='\t', quote=F)  
          q()
    2. Update family and individual IDs using the following commands:
       1. If the IID was not changed from SENTRIX format use:  
          plink --bfile original-initials-eur-3-het --update-ids original-initials-eur-king-3updateids.txt --make-bed --out original-initials-eur-king-3-ids
       2. If the IID was changed from SENTRIX format use:  
          plink --bfile original-initials-eur-3-het --update-ids original-initials-eur-king-3updateids.txt-sentrix --make-bed --out original-initials-eur-king-3-ids
    3. Update paternal and maternal IDs using the following commands:
       1. If the IID was not changed from SENTRIX format use:  
          plink --bfile original-initials-eur-king-3-ids --update-parents original-initials-eur-king-3updateparents.txt --make-bed --out original-initials-eur-king-3-parents
    4. If the IID was changed from SENTRIX format use:  
       plink --bfile original-initials-eur-king-3-ids --update-parents original-initials-eur-king-3updateparents.txt-sentrix --make-bed --out original-initials-eur-king-3-parents

### Identified relationships:

* + 1. Look at the original-initials-eur-king-3-slurm.txt file and record the number of various between and within family relationships that KING identified/inferred in comparison to MoBa (during the related analysis).

### YOB and Sex check

In the “resources” folder in DATA/DURABLE, there are files named “yob.txt” and “sex.txt” that contains the year of birth of MoBa participants (the three columns in the file are: FID, IID and year-of-birth) and the sex of MoBa participants (the three columns in the file are: FID, IID and sex), respectively. Copy these files to your working directory and run the commands below.

* + 1. YOB check  
       ./match.pl -f yob.txt -g original-initials-eur-king-3-parents.fam -k 2 -l 2 -v 3 > original-initials-eur-king-3-children-yob.txt  
       ./match.pl -f yob.txt -g original-initials-eur-king-3-children-yob.txt -k 2 -l 3 -v 3 > original-initials-eur-king-3-children-fathers-yob.txt  
       ./match.pl -f yob.txt -g original-initials-eur-king-3-children-fathers-yob.txt -k 2 -l 4 -v 3 > original-initials-eur-king-3-yob.txt  
       rm original-initials-eur-king-3-children-yob.txt original-initials-eur-king-3-children-fathers-yob.txt  
       awk ‘{if ($7<$8 || $7<$9) print $0, “PROBLEM”; else print $0, “OK”}’ original-initials-eur-king-3-yob.txt > original-initials-eur-king-3-yob-check.txt  
       awk ‘$10==”PROBLEM” {print $0}’ original-initials-eur-king-3-yob-check.txt > original-initials-eur-king-3-yob-problem.txt
    2. Sex check  
       ./match.pl -f original-initials-eur-king-3-parents.fam -g original-initials-eur-king-3-parents.fam -k 2 -l 3 -v 5 > original-initials-eur-king-3-father-sex.txt  
       ./match.pl -f original-initials-eur-king-3-parents.fam -g original-initials-eur-king-3-father-sex.txt -k 2 -l 4 -v 5 > original-initials-eur-king-3-sex.txt  
       rm original-initials-eur-king-3-father-sex.txt  
       awk ‘{if ($7==2 || $8==1) print $0, “PROBLEM”; else print $0, “OK”}’ original-initials-eur-king-3-sex.txt > original-initials-eur-king-3-sex-check.txt  
       awk ‘$10==”PROBLEM” {print $0}’ original-initials-eur-king-3-sex-check.txt > original-initials-eur-king-3-sex-problem.txt
    3. Check the number of problematic families. Use the following commands to get the number:  
       wc -l original-initials-eur-king-3-yob-problem.txt  
       wc -l original-initials-eur-king-3-sex-problem.txt

### Examine the relationships within families (kin file)

* + 1. Identify any instances where the inferred relationships do not match those reported in MoBa:  
       awk ‘$16>0 {print $0}’ original-initials-eur-king-3.kin > original-initials-eur-king-3.kin-errors  
       ./match.pl -f /tsd/p697/data/durable/projects/moba\_qc\_imputation/resources/genotyped\_pedigree.txt -g original-initials-eur-king-3.kin-errors -k 4 -l 2 -v 8 > original-initials-eur-king-3.kin-errors-role1  
       ./match.pl -f /tsd/p697/data/durable/projects/moba\_qc\_imputation/resources/genotyped\_pedigree.txt -g original-initials-eur-king-3.kin-errors-role1 -k 4 -l 3 -v 8 > original-initials-eur-king-3.kin-errors  
       rm original-initials-eur-king-3.kin-errors-role1  
       wc -l original-initials-eur-king-3.kin-errors
       1. Identify any inferred duplicates/MZ twins that were unexpected:  
          awk ‘$15==”Dup/MZ” {print $0}’ original-initials-eur-king-3.kin-errors > original-initials-eur-king-3.kin-errors-MZ  
          wc -l original-initials-eur-king-3.kin-errors-MZ
       2. Identify any inferred parent-offspring relationships that were unexpected:  
          awk ‘$15==”PO” {print $0}’ original-initials-eur-king-3.kin-errors > original-initials-eur-king-3.kin-errors-PO  
          wc -l original-initials-eur-king-3.kin-errors-PO
       3. Identify any inferred full siblings that were unexpected:  
          awk ‘$15==”FS” {print $0}’ original-initials-eur-king-3.kin-errors > original-initials-eur-king-3.kin-errors-FS  
          wc -l original-initials-eur-king-3.kin-errors-FS
       4. Identify any inferred second degree relatives that were unexpected:  
          awk ‘$15==”2nd” {print $0}’ original-initials-eur-king-3.kin-errors > original-initials-eur-king-3.kin-errors-2nd  
          wc -l original-initials-eur-king-3.kin-errors-2nd
       5. Identify any inferred third degree relatives that were unexpected:  
          awk ‘$15==”3rd” {print $0}’ original-initials-eur-king-3.kin-errors > original-initials-eur-king-3.kin-errors-3rd  
          wc -l original-initials-eur-king-3.kin-errors-3rd
       6. Identify any inferred fourth degree relatives that were unexpected:  
          awk ‘$15==”4th” {print $0}’ original-initials-eur-king-3.kin-errors > original-initials-eur-king-3.kin-errors-4th  
          wc -l original-initials-eur-king-3.kin-errors-4th
       7. Identify any inferred unrelated individuals that were unexpected:  
          awk ‘$15==”UN” {print $0}’ original-initials-eur-king-3.kin-errors > original-initials-eur-king-3.kin-errors-UN  
          wc -l original-initials-eur-king-3.kin-errors-UN

### Examine the relationships between families (kin0 file)

* + 1. Identify any instances where the inferred relationships do not match those reported in MoBa:  
       awk ‘{if ($14==”Dup/MZ” || $14==”PO” || $14==”FS”) print $0}’ original-initials-eur-king-3.kin0 > original-initials-eur-king-3.kin0-errors  
       ./match.pl -f /tsd/p697/data/durable/projects/moba\_qc\_imputation/resources/genotyped\_pedigree.txt -g original-initials-eur-king-3.kin0-errors -k 4 -l 2 -v 8 > original-initials-eur-king-3.kin0-errors-role1  
       ./match.pl -f /tsd/p697/data/durable/projects/moba\_qc\_imputation/resources/genotyped\_pedigree.txt -g original-initials-eur-king-3.kin0-errors-role1 -k 4 -l 4 -v 8 > original-initials-eur-king-3.kin0-errors  
       rm original-initials-eur-king-3.kin0-errors-role1  
       wc -l original-initials-eur-king-3.kin0-errors
       - 1. Identify any inferred duplicates/MZ twins that were unexpected:  
            awk ‘$14==”Dup/MZ” {print $0}’ original-initials-eur-king-3.kin0-errors > original-initials-eur-king-3.kin0-errors-MZ  
            wc -l original-initials-eur-king-3.kin0-errors-MZ
         2. Identify any inferred parent-offspring relationships that were unexpected:  
            awk ‘$14==”PO” {print $0}’ original-initials-eur-king-3.kin0-errors > original-initials-eur-king-3.kin0-errors-PO  
            wc -l original-initials-eur-king-3.kin0-errors-PO
         3. Identify any inferred full siblings that were unexpected:  
            awk ‘$14==”FS” {print $0}’ original-initials-eur-king-3.kin0-errors > original-initials-eur-king-3.kin0-errors-FS  
            wc -l original-initials-eur-king-3.kin0-errors-FS

### Plot the relationships

* + 1. Make histograms of the estimated kinship coefficients between families
       1. Create file that is smaller for reading into R  
          awk ‘{print $2, $4, $19}’ original-initials-eur-king-3.ibs0 > original-initials-eur-king-3.ibs0\_hist
       2. Run plot-kinship-histogram.R according to the instructions in $GITHUB/lib/README-for-plot-kinship-histograms.txt. Please specify the following: input: original-initials-eur-king-3.ibs0\_hist and output: original-initials-eur-king-3-hist  
          Example: Rscript $GITHUB/lib/plot-kinship-histogram.R m12good-ec-eur-king-3.ibs0\_hist m12good-ec-eur-king-3-hist
       3. Remove the file used to plot the kinship histogram  
          rm original-initials-eur-king-3.ibs0\_hist
       4. Please post the graph on slack, copy it to the export folder for module II, and record the name of the plot in the google sheet.
    2. Make plots to visualize the between and within family relationships as reported by MoBa vs inferred by KING
    3. Please run the create-relplot.sh script that uses the relplot R script from KING, and generates a png file with four or two merged subplots with optionally customized legend positions. Usage information is on github (<https://github.com/norment/moba_qc_imputation/tree/master/tools#create-relplotsh>).  
       Usage: sh $GITHUB/tools/create-relplot.sh r\_relplot tag [legendpos1 legendpos2 legendpos3 legendpos4]  
       Arguments:  
       r\_relplot - R script file for relplot from KING  
       tag - a tag of data shown in the titles of the plots  
       legendpos1 - legend position of plot 1: topleft, topright, bottomleft, bottomright  
       legendpos2 - legend position of plot 2: topleft, topright, bottomleft, bottomright  
       legendpos3 - legend position of plot 3: topleft, topright, bottomleft, bottomright  
       legendpos4 - legend position of plot 4: topleft, topright, bottomleft, bottomright  
       Example: sh create-relplot.sh rotterdam1-yc-eur-king-3\_relplot.R "Rotterdam1 EUR, round 3" topright bottomright topright bottomright
    4. Post the plots on slack, copy the plot file to the export folder for plots of Module II, and record the plot name in the google sheet for report production.

### Fix within and between family issues

* + 1. Let Elizabeth know once you have completed the above steps and she will investigate the pedigree errors and create the update and remove files.
    2. Remove individuals with impossible relationships and update family and individual ids.  
       plink --bfile original-initials-eur-king-3-parents --remove original-initials-eur-king-3-unexpected-relationships.txt --update-ids original-initials-eur-king-3-fix-ids.txt --make-bed --out original-initials-eur-king-3-fix-ids
    3. Update paternal and maternal ids.  
       plink --bfile original-initials-eur-king-3-fix-ids --update-parents original-initials-eur-king-3-fix-parents.txt --make-bed --out original-initials-eur-king-3-fix-parents

### Identify any pedigree issues

* + 1. Create the covariate file
       1. Copy the age.txt file to your working directory.
       2. Update the FID to the match those updated during the above KING analysis.  
          ./match.pl -f original-initials-eur-king-3-fix-parents.fam -g age.txt -k 2 -l 2 -v 1 | awk ‘$4!=”-” {print $4, $2, $3}’ > original-initials-eur-king-3-fix-parents.cov  
          sed -i ‘1 i FID IID Age’ original-initials-eur-king-3-fix-parents.cov
    2. Run KING build using the following command:  
       /cluster/projects/p697/projects/moba\_qc\_imputation/software/king225 -b original-initials-eur-king-3-fix-parents.bed --build --degree 2 --prefix original-initials-eur-king-3.5 > original-initials-eur-king-3.5-slurm.txt
    3. Identify any issues
       1. Go through the build log file and see if there were any pedigree errors reported.
       2. See if an update ids or parents file was produced.
    4. Let Elizabeth know if there were any issues and she will investigate what is going on.
    5. Remove individuals with impossible relationships and update family and individual ids.
       1. Remove individuals with impossible relationships and update family and individual ids.  
          plink --bfile original-initials-eur-king-3-fix-parents --remove original-initials-eur-king-3.5-unexpected-relationships.txt --update-ids original-initials-eur-king-3.5-fix-ids.txt --make-bed --out original-initials-eur-king-3.5-fix-ids
       2. Update paternal and maternal ids.  
          plink --bfile original-initials-eur-king-3.5-fix-ids --update-parents original-initials-eur-king-3.5-fix-parents.txt --make-bed --out original-initials-eur-king-3.5-fix-parents
       3. If you had individuals removed, get the IDs of the removed individuals and add them to the IDs of individuals who were removed previously:  
          ./match.pl -f original-initials-eur-king-3.5-fix-parents.fam -g original-initials-eur-king-3-fix-parents.fam -k 2 -l 2 -v 1 | awk ‘$7==”-” {print $1,$2,”pedigree and known relatedness, round 3”}’ >> original-initials-eur-3-removed-individuals.txt

## IBD estimation

### Prune the data and remove long stretches of LD

* + 1. Prune the data  
       plink --bfile original-initials-eur-king-3.5-fix-parents --indep-pairwise 3000 1500 0.1 --out original-initials-eur-king-3.5-indep
       1. If needed, repeat the pruning to remove residual LD, until there are about 100,000 SNPs left.
    2. Long LD regions
       1. Copy the file containing the list of long LD regions (Build37) from the “resources” folder to your working directory (name of the file is “high-ld.txt”)
       2. Extract pruned SNPs and remove long stretches of LD  
          plink --bfile original-initials-eur-king-3.5-fix-parents --extract original-initials-eur-king-3.5-indep.prune.in --make-set high-ld.txt --write-set --out original-initials-eur-king-3.5-highld  
          plink --bfile original-initials-eur-king-3.5-fix-parents --extract original-initials-eur-king-3.5-indep.prune.in --exclude original-initials-eur-king-3.5-highld.set --make-bed --out original-initials-eur-king-3.5-trimmed
  1. Record the number of SNPs in the following files  
     wc -l original-initials-eur-king-3.5-indep.prune.in  
     cat original-initials-eur-king-3.5-highld.set | grep -v hild | grep -v END | grep -v "^$" | wc -l  
     wc -l original-initials-eur-king-3.5-trimmed.bim

### Run IBD calculations

* + 1. Copy the original-initials-eur-king-3.5-trimmed.bed, original-initials-eur-king-3.5-trimmed.bim, and original-initials-eur-king-3.5-trimmed.fam files to your working directory on Colossus.
    2. Go to your working directory on Colossus and run the $GITHUB/jobs/IBD.job to calculate the PI\_HAT in PLINK (create the genome file) and plot histograms of PI\_HAT as well as Z0 vs Z1 plot. Define “fin” and “fout” and “tag” environmental variables as shown below, adjusting the names to reflect the names of your files (e.g. “original-initials” needs to become “moba12good-ec”), and then submit the IBD job. Please remember to use “plot-PLINK” file (in “resources” folder) to use the correct tag for your batch. “Population” refers to the core subpopulation you are QC-ing.  
       export fin=original-initials-eur-king-3.5-trimmed  
       export fout=original-initials-eur-king-3-ibd  
       export tag="tag population, round 3"  
       sbatch $GITHUB/jobs/IBD.job
    3. After the job is completed, remove the files you used to calculate and plot PI\_HAT and move the output to your working directory in DATA/DURABLE  
       rm original-initials-eur-king-3.5-trimmed.\*  
       mv original-initials-eur-king-3-ibd.\*  
       /tsd/p697/data/durable/projects/moba\_qc\_imputation/initials  
       mv \*.png /tsd/p697/data/durable/projects/moba\_qc\_imputation/initials

### Check IBD patterns are as expected.

* + 1. Please review the IBD plots and post them on slack.

#### Make a list of those who are related, but share too little/too much: awk ‘$5==“PO” && $10<0.4 || $5==“PO” && $10>0.6 {print $0}’ original-initials-eur-king-3-ibd.genome > original-initials-eur-king-3-ibd-bad-parents.txt awk ‘$5==“FS” && $10<0.4 || $5==“FS” && $10>0.6 {print $0}’ original-initials-eur-king-3-ibd.genome > original-initials-eur-king-3-ibd-bad-siblings.txt awk ‘$5==“HS” && $10<0.15 || $5==“HS” && $10>0.35 {print $0}’ original-initials-eur-king-3-ibd.genome > original-initials-eur-king-3-ibd-bad-half-siblings.txt awk ‘$5!=“PO” && $5!=”FS” && $5!=”HS” && $10>0.15 {print $0}’ original-initials-eur-king-3-ibd.genome > original-initials-eur-king-3-ibd-bad-unrelated.txt cat original-initials-eur-king-3-ibd-bad-unrelated.txt original-initials-eur-king-3-ibd-bad-parents.txt original-initials-eur-king-3-ibd-bad-siblings.txt original-initials-eur-king-3-ibd-bad-half-siblings.txt > original-initials-eur-king-3-ibd-bad-relatedness.txt rm original-initials-eur-king-3-ibd-bad-unrelated.txt original-initials-eur-king-3-ibd-bad-parents.txt original-initials-eur-king-3-ibd-bad-siblings.txt original-initials-eur-king-3-ibd-bad-half-siblings.txt

* + 1. Merge pairwise bad relatedness list with the KING inferred relatedness type  
       awk ‘{print $2,$3,$15}’ original-initials-eur-king-3.kin > original-initials-eur-king-3.kin-RT  
       awk ‘{print $2,$4,$14}’ original-initials-eur-king-3.kin0 > original-initials-eur-king-3.kin0-RT  
       cat original-initials-eur-king-3.kin-RT original-initials-eur-king-3.kin0-RT > original-initials-eur-king-3.RT  
       rm original-initials-eur-king-3.kin-RT original-initials-eur-king-3.kin0-RT  
       R  
       bad <- read.table(‘original-initials-eur-king-3-ibd-bad-relatedness.txt’,h=T,colClasses=”character”)  
       bad\_match <- subset(bad, FID1==FID2)  
       bad\_nonmatch <- subset(bad, FID1!=FID2)  
       rm(bad)  
       kin <- read.table(‘original-initials-eur-king-3.RT’,h=T)  
       colnames(kin) <- c(“IID1”,“IID2”,“InfType”)  
       bad\_kin1 <- merge(bad\_match, kin, by=c(“IID1”, “IID2”))  
       bad\_kin2 <- merge(bad\_match, kin, by.x=c(“IID1”, “IID2”), by.y=c(“IID2”, “IID1”))  
       bad\_kin <- rbind(bad\_kin1, bad\_kin2)  
       rm(bad\_kin1, bad\_kin2)  
       table(bad\_kin$InfType)  
       bad\_kin <- bad\_kin[,c(3,1,4,2,5:15)]  
       write.table(bad\_kin, ‘original-initials-eur-king-3-ibd-bad-relatedness.txt-InfType’,row.names=F, col.names=T, sep=’\t’, quote=F)  
       rm(bad\_kin)  
       id1 <- data.frame(bad\_nonmatch[,2])  
       colnames(id1) <- “IID”  
       id2 <- data.frame(bad\_nonmatch[,4])  
       colnames(id2) <- “IID”  
       ids <- rbind(id1, id2)  
       rm(id1, id2)  
       length(unique(ids$IID))  
       freq <- data.frame(table(ids$IID))  
       rm(ids)  
       colnames(freq) <- c(“IID1”, “Freq1”)  
       id1 <- merge(bad\_nonmatch, freq, by=”IID1”)  
       colnames(freq) <- c(“IID2”, “Freq2”)  
       bad\_nonmatch <- merge(id1, freq, by=”IID2”)  
       rm(id1, freq)  
       bad\_nonmatch <- bad\_nonmatch[,c(3,2,4,1,5:16)]  
       write.table(bad\_nonmatch, ‘original-initials-eur-king-3-ibd-bad-relatedness.txt-Freq’,row.names=F, col.names=T, sep=’\t’, quote=F)  
       rm(bad\_nonmatch)  
       q()
    2. Add individual missingness rate to the bad relatedness files  
       plink --bfile original-initials-eur-king-3-fix-parents --missing --out original-initials-eur-king-3-missing  
       ./match.pl -f original-initials-eur-king-3-missing.imiss -g original-initials-eur-king-3-ibd-bad-relatedness.txt-InfType -k 2 -l 2 -v 6 > original-initials-eur-king-3-ibd-bad-relatedness.txt-InfType-call-rate1  
       ./match.pl -f original-initials-eur-king-3-missing.imiss -g original-initials-eur-king-3-ibd-bad-relatedness.txt-InfType-call-rate1 -k 2 -l 4 -v 6 > original-initials-eur-king-3-ibd-bad-relatedness.txt-InfType-call-rate  
       ./match.pl -f original-initials-eur-king-3-missing.imiss -g original-initials-eur-king-3-ibd-bad-relatedness.txt-Freq -k 2 -l 2 -v 6 > original-initials-eur-king-3-ibd-bad-relatedness.txt-Freq-call-rate1  
       ./match.pl -f original-initials-eur-king-3-missing.imiss -g original-initials-eur-king-3-ibd-bad-relatedness.txt-Freq-call-rate1 -k 2 -l 4 -v 6 > original-initials-eur-king-3-ibd-bad-relatedness.txt-Freq-call-rate  
       rm original-initials-eur-king-3-ibd-bad-relatedness.txt-InfType-call-rate1 original-initials-eur-king-3-ibd-bad-relatedness.txt-Freq-call-rate1
    3. Let Elizabeth know once you have completed the above steps and she will investigate the “bad” individuals and make a list of individuals to remove (and files to recode the family relationships if needed).

### Remove individuals sharing more or less IBD than expected

* + 1. Remove individuals with unexpected IBD patterns and update parents to accurately reflect relationships. The updating of parental IDs will not be needed for all batches. An example of when it is needed is to update HS so they have different numeric parental codes, as plink interprets ‘0’ as an ID if there is one parental ID for an individual.  
       plink --bfile original-initials-eur-king-3.5-fix-parents --update-parents original-initials-eur-king-3-bad-relatedness-update-parents.txt --remove original-initials-eur-king-3-bad-relatedness-low-call-rate.txt --make-bed --out original-initials-eur-king-3-ibd-clean
    2. Record how many individuals were removed (from numbers in the log file).
    3. Add the IDs of the individuals removed to the list of those who were removed previously:  
       awk ‘{print $1,$2,”IBD, round 3”}’ original-initials-eur-king-3-bad-relatedness-low-call-rate.txt >> original-initials-eur-3-removed-individuals.txt

## Cryptic relatedness

In this section, we aim to remove individuals who share too much kinship with too many people.

* 1. Create files with (1) counts of individuals with whom each individual shares >=2.5% kinship and (2) sum of all kinship coefficients >=2.5% per individual.
     1. Copy “cryptic.sh” file from “scripts” folder to your working directory and make the file executable.  
        chmod +x cryptic.sh
     2. Run the cryptic.sh script using the ibs0 output from KING.  
        ./cryptic.sh original-initials-eur-king-3.ibs0 original-initials-eur-cryptic-3
  2. Create plots
     1. Run the $GITHUB/lib/plot-cryptic.R script. See the “README-for-cryptic-plot” file for usage instructions.  
        For input1, please use “original-initials-eur-cryptic-3-kinship-sum.txt” file  
        For input2, please use “original-initials-eur-cryptic-3-counts.txt” file  
        For tag, please use the tag from plot-PLINK file  
        For output, please use original-initials-eur-cryptic-3  
        Example command: Rscript $GITHUB/lib/plot-cryptic.R m24-tz-eur-cryptic-3-kinship-sum.txt m24-tz-eur-cryptic-3-counts.txt "M24 EUR" m24-tz-eur-cryptic-3
     2. Please post your plot on slack, where we’ll determine what threshold would fit the data in your batch the best.
  3. Remove outliers
     1. Identify cryptic relatedness outliers. If your threshold is, for example, 15 for the sum of kinship, the example commands would be:  
        awk ‘$2>15 {print $1}’ original-initials-eur-cryptic-3-kinship-sum.txt > original-initials-eur-cryptic-3-sum-remove  
        ./match.pl -f original-initials-eur-king-3-ibd-clean.fam -g original-initials-eur-cryptic-3-sum-remove -k 2 -l 1 -v 1 | awk ‘{print $2, $1}’ > original-initials-eur-cryptic-3-sum-remove.txt
     2. Remove cryptic relatedness outliers.  
        plink --bfile original-initials-eur-king-3-ibd-clean --remove original-initials-eur-cryptic-3-sum-remove.txt --make-bed --out original-initials-eur-cryptic-clean-3
     3. Please make sure that the number of individuals removed (as shown in the log file) corresponds or is in line with the number of individuals in original-initials-eur-cryptic-3-sum-remove.txt file.  
        wc -l original-initials-eur-cryptic-3-sum-remove.txt
     4. Please record the number of individuals removed due to cryptic relatedness.
     5. If you had individuals removed, get the IDs of the removed individuals and add them to the IDs of individuals who were removed previously:  
        ./match.pl -f original-initials-eur-king-3-ibd-clean.fam -g original-initials-eur-cryptic-clean-3.fam -k 2 -l 2 -v 1 | awk ‘$7==”-” {print $1,$2,”cryptic relatedness, round 3”}’ >> original-initials-eur-3-removed-individuals.txt

## Mendelian errors

* 1. Remove families with more than 5% Mendel errors and SNPs with more than 1% of Mendel errors and zeros out the other minor Mendel errors.
     1. If your fam file contains no individuals with unknown sex run the below command then move to step 12.2  
        plink --bfile original-initials-eur-cryptic-clean-3 --me 0.05 0.01 --set-me-missing --mendel-duos --make-bed --out original-initials-eur-me-clean-sex-3
     2. If your fam file contains individuals with unknown sex, use the PLINK inferred sex for people whose sex is missing.  
        awk ‘$3==0 {print $1,$2,$4}’ original-initials-eur-2-sexcheck.sexcheck > original-initials-eur-3-sex-me.txt  
        plink --bfile original-initials-eur-cryptic-clean-3 --update-sex original-initials-eur-3-sex-me.txt --me 0.05 0.01 --set-me-missing --mendel-duos --make-bed --out original-initials-eur-me-clean-3
     3. Restore the sex that was before update for ME check purposes:  
        awk ‘{print $1,$2,0}’ original-initials-eur-3-sex-me.txt > original-initials-eur-3-sex-me-back.txt
     4. plink --bfile original-initials-eur-me-clean-3 --update-sex original-initials-eur-3-sex-me-back.txt --make-bed --out original-initials-eur-me-clean-sex-3
  2. Record the number of families removed, the number of SNPs removed and the number of Mendelian errors zero-ed out (all these numbers can be obtained from the plink log file of step 20.1).
  3. If you had individuals (families) removed, get the IDs of the removed individuals and add them to the IDs of individuals who were removed previously:  
     ./match.pl -f original-initials-eur-me-clean-sex-3.fam -g original-initials-eur-cryptic-clean-3.fam -k 2 -l 2 -v 1 | awk ‘$7==”-” {print $1,$2,”Mendelian-error, round 3”}’ >> original-initials-eur-3-removed-individuals.txt
  4. If you had SNPs removed, add their IDs to those that were removed previously:  
     ./match.pl -f original-initials-eur-me-clean-sex-3.bim -g original-initials-eur-cryptic-clean-3.bim -k 2 -l 2 -v 1 | awk ‘$7==”-” {print $2,”Mendelian-error, round 3”}’ >> original-initials-eur-3-bad-snps.txt

## PCA with 1KG

### Prune the data and remove long stretches of LD

* + 1. Prune the data  
       plink --bfile original-initials-eur-me-clean-sex-3 --indep-pairwise 3000 1500 0.1 --out original-initials-eur-me-clean-sex-3-indep
       1. If needed, repeat the pruning to remove residual LD, until there are about 100,000 SNPs left.
    2. Extract the set of pruned SNPs  
       plink --bfile original-initials-eur-me-clean-sex-3 --extract original-initials-eur-me-clean-sex-3-indep.prune.in --make-bed --out original-initials-eur-me-clean-sex-3-pruned
    3. Long LD regions
       1. Copy the file containing the list of long LD regions (Build37) from the “resources” folder to your working directory (name of the file is “high-ld.txt”).
       2. Create high LD set  
          plink --bfile original-initials-eur-me-clean-sex-3-pruned --make-set high-ld.txt --write-set --out original-initials-eur-me-clean-sex-3-highld
       3. Exclude SNPs in high LD set  
          plink --bfile original-initials-eur-me-clean-sex-3-pruned --exclude original-initials-eur-me-clean-sex-3-highld.set --make-bed --out original-initials-eur-me-clean-sex-3-trimmed

### Identify SNPs overlapping with 1KG

* + 1. Copy the 1KG PLINK bfiles from the “resources” folder in DATA/DURABLE to your working directory (1kg.bed,1kg.bim and 1kg.fam); these files have the strand ambiguous SNPs already removed.
    2. Run the following commands to identify the overlapping SNPs  
       cut -f2 1kg.bim | sort -s > 1kg.bim.sorted  
       cut -f2 original-initials-eur-me-clean-sex-3-trimmed.bim | sort -s > original-initials-eur-me-clean-sex-3-trimmed.bim.sorted  
       join 1kg.bim.sorted original-initials-eur-me-clean-sex-3-trimmed.bim.sorted > original-initials-eur-me-clean-sex-3-1kg-snps.txt  
       rm original-initials-eur-me-clean-sex-3-trimmed.bim.sorted
    3. Record the number of SNPs you have common in your batch and in 1KG:  
       wc -l original-initials-eur-me-clean-sex-3-1kg-snps.txt

### Merge with the 1KG dataset

* + 1. Extract the overlapping SNPs
       1. In your batch:  
          plink --bfile original-initials-eur-me-clean-sex-3-trimmed --extract original-initials-eur-me-clean-sex-3-1kg-snps.txt --make-bed --out original-initials-eur-me-clean-sex-3-1kg-common
       2. In the 1KG dataset:  
          plink --bfile 1kg --extract original-initials-eur-me-clean-sex-3-1kg-snps.txt --make-bed --out 1kg-original-initials-eur-me-clean-sex-3-common
    2. Merge the bfiles  
       plink --bfile original-initials-eur-me-clean-sex-3-1kg-common --bmerge 1kg-original-initials-eur-me-clean-sex-3-common --make-bed --out original-initials-eur-me-clean-sex-3-1kg-merged
       1. If you have SNPs that have 3+ alleles, flip those alleles in 1kg data and merge again. To flip, run the following command:  
          plink --bfile 1kg-original-initials-eur-me-clean-sex-3-common --flip original-initials-eur-me-clean-sex-3-1kg-merged-merge.missnp --make-bed --out 1kg-original-initials-eur-me-clean-sex-3-common-flip  
          To merge again, run the following command:  
          plink --bfile original-initials-eur-me-clean-sex-3-1kg-common --bmerge 1kg-original-initials-eur-me-clean-sex-3-common-flip --make-bed --out original-initials-eur-me-clean-sex-3-1kg-second-merged
       2. If you still have SNPs with 3+ alleles after merging with flipped data, remove those SNPs from both 1KG and MoBa data. To remove the SNPs with 3+ alleles from MoBa and from 1kg, run the following commands:  
          plink --bfile original-initials-eur-me-clean-sex-3-1kg-common --exclude original-initials-eur-me-clean-sex-3-1kg-second-merged-merge.missnp --make-bed --out original-initials-eur-me-clean-sex-3-1kg-common-clean  
          plink --bfile 1kg-original-initials-eur-me-clean-sex-3-common-flip --exclude original-initials-eur-me-clean-sex-3-1kg-second-merged-merge.missnp --make-bed --out 1kg-original-initials-eur-me-clean-sex-3-common-flip-clean  
          To merge again, run the following command:  
          plink --bfile original-initials-eur-me-clean-sex-3-1kg-common-clean --bmerge 1kg-original-initials-eur-me-clean-sex-3-common-flip-clean --make-bed --out original-initials-eur-me-clean-sex-3-1kg-clean-merged
    3. Record how many SNPs you are going to use for PCA:  
       wc -l original-initials-eur-me-clean-sex-3-1kg-clean-merged.bim

### PCA

* + 1. Copy the populations.txt file from “resources” folder on DATA/DURABLE to your working directory. populations.txt is a text file containing the population, based on which PLINK will calculate the main PCs, in this case it will be just one word “parent”).
    2. Create the original-initials-eur-me-clean-sex-3-fam-populations.txt – a text file with 3 columns: family ID, individual ID and so-called population, in this case it will be “parent” for unrelated individuals and “child” for related individuals. To create this file, do the following:  
       awk ‘{if($3==0 && $4==0) print $1,$2,”parent”; else print $1,$2,”child”}’ original-initials-eur-me-clean-sex-3-1kg-merged.fam > original-initials-eur-me-clean-sex-3-fam-populations.txt
       1. If you did the flip, use original-initials-eur-me-clean-sex-3-1kg-second-merged.fam
       2. If you had to remove SNPs after the flip, use original-initials-eur-me-clean-sex-3-1kg-clean-merged.fam
    3. Run the PCA  
       plink --bfile original-initials-eur-me-clean-sex-3-1kg-merged --pca --within original-initials-eur-me-clean-sex-3-fam-populations.txt --pca-clusters populations.txt --out original-initials-eur-me-clean-sex-3-1kg-pca
       1. If your data had SNPs with 3+ alleles and you did flip+merge steps, use the original-initials-eur-me-clean-sex-3-1kg-second-merged bfiles
       2. If your data had SNPs with 3+ alleles after flip+merge steps and you did remove+merge steps, then use the original-initials-eur-me-clean-sex-3-1kg-clean-merged bfiles
       3. If your batch is large and you need to run the PCA on Colossus, copy your PLINK input files into Colossus and use the PCA\_POP.job

### Plot PCs

* + 1. The individuals in 1kg.fam file are not ordered according to their population. But in order to assign colors during plotting, we need them to be in order. Thus, we will divide the file containing the PCs into “original-eur” and “1kg” portions, sort the “1kg” portion and combine the two together for plotting. This can be achieved with the following commands:  
       sort -k2 original-initials-eur-me-clean-sex-3-1kg-pca.eigenvec > original-initials-eur-me-clean-sex-3-1kg-pca-sorted  
       head -n “a” original-initials-eur-me-clean-sex-3-1kg-pca-sorted > original-initials-eur-me-clean-sex-3-1kg-pca-1  
       NB the “a” in the “head” command is the number of individuals in the “original” PLINK files that were merged with 1KG (it is the number of individuals in original-initials-eur-me-clean-sex-3-trimmed.fam file)  
       tail -n 1083 original-initials-eur-me-clean-sex-3-1kg-pca-sorted | sort -k2 > 1kg-initials-eur-3-pca-1  
       NB 1083 is the number of individuals in 1kg.fam  
       cat original-initials-eur-me-clean-sex-3-1kg-pca-1 1kg-initials-eur-3-pca-1 > original-initials-eur-me-clean-sex-3-1kg-pca
    2. Plot your batch with the 1KG dataset.  
       The code for plotting is located in $GITHUB/lib/plot-pca-with-1kg.R. Instructions are in $GITHUB/lib/README.md
       1. Please remember to use the “plot-PLINK” file in “resources” folder to assign the harmonized tag to the plots of your batch.  
          Please specify the “outprefix” as “original-initials-eur-me-clean-sex-3-1kg”.
       2. When you’ve made the plots, please post them on slack and place a copy in the export folder for Module II.
    3. If needed, select a cleaner subsample using the script named “$GITHUB/lib/select-subsamples-on-pca.R”. Please name the customfile as “original-initials-eur-pca-core-select-4-custom.txt” and the “outprefix” as “original-initials-eur-selection-4”.  
       For examples of <customfile> see in $GITHUB/config folder.
       1. Please note that here we select only EUR subsample (in other words, we tighten the previous selection). Thus, the customfile will have only EUR section. Your customfile may look like this:  
          eur\_zoom\_threshold: PC1> -0.01  
          eur\_draw\_threshold: PC1>0  
          eur\_legend\_position: bottomleft
       2. In this example, the selection is done based on PC1 only. Please add PC2 and/or other PCs as required.
    4. Once done, please copy the plots to /tsd/p697/data/durable/projects/moba\_qc\_imputation/export/Module\_II\_Plots folder.
    5. Please record the names of the plots in MoBa\_QC\_numbers sheet.

### Remove ancestry outliers

* + 1. Use the following command to remove PC outliers:  
       plink --bfile original-initials-eur-me-clean-sex-3 --keep original-initials-eur-selection-4-core-subsample-eur.txt --make-bed --out original-initials-eur-4-keep
    2. Please record the number of individuals kept in MoBa\_QC\_numbers sheet (you can see the number in the log file of the above PLINK command).
    3. Add the IDs of individuals removed to those who were removed previously:  
       ./match.pl -f original-initials-eur-4-keep.fam -g original-initials-eur-me-clean-sex-3.fam -k 2 -l 2 -v 1 | awk ‘$7==”-” {print $1,$2,”PCA-with-1kg-round-3”}’>> original-initials-eur-3-removed-individuals.txt
    4. If there are no outliers in this step, then take the original-initials-eur-me-clean-sex-3 biles into step 22.

## PCA without 1KG

### Prune and remove long stretches of LD

* + 1. Prune the data
       1. If there were outliers:  
          plink --bfile original-initials-eur-4-keep --indep-pairwise 3000 1500 0.1 --out original-initials-eur-4-keep-indep
       2. If there were no outliers in step 21, then run the following command:  
          plink --bfile original-initials-eur-me-clean-sex-3 --indep-pairwise 3000 1500 0.1 --out original-initials-eur-4-keep-indep
       3. Check the number of remaining SNPs after pruning:  
          wc -l original-initials-eur-4-keep-indep.prune.in
          1. If the number is substantially larger than 100K, repeat the pruning until there are about 100K SNPs left.
    2. Remove long stretches of LD  
       plink --bfile original-initials-eur-4-keep --extract original-initials-eur-4-keep-indep.prune.in --make-set high-ld.txt --write-set --out original-initials-eur-4-keep-highld  
       plink --bfile original-initials-eur-4-keep --extract original-initials-eur-4-keep-indep.prune.in --exclude original-initials-eur-4-keep-highld.set --make-bed --out original-initials-eur-4-trimmed
    3. Record the number of SNPs in original-initials-eur-4-trimmed.bim  
       wc -l original-initials-eur-4-trimmed.bim

### PCA

* + 1. Copy the populations.txt file from “resources” folder on DATA/DURABLE to your working directory. populations.txt is a text file containing the population, based on which PLINK will calculate the main PCs, in this case it will be just one word “parent”).
    2. Create original-initials-eur-4-populations.txt – a text file with 3 columns: family ID, individual ID and so-called population, in this case it will “parent” for unrelated individuals and “child” for related individuals.  
       awk ‘{if($3==0 && $4==0) print $1,$2,”parent”; else print $1,$2,”child”}’ original-initials-eur-4-trimmed.fam > original-initials-eur-4-populations.txt
    3. Run PCA in PLINK  
       plink --bfile original-initials-eur-4-trimmed --pca --within original-initials-eur-4-populations.txt --pca-clusters populations.txt --out original-initials-eur-4-pca
       1. If your batch is large and you need to run the PCA on Colossus, copy your PLINK input files into Colossus and use the PCA\_POP.job

### Plot PCs

* + 1. Identify founders and non-founders  
       awk ‘{if($3==0 && $4==0) print $1,$2,”black”; else print $1,$2,”red” }’ original-initials-eur-4-keep.fam > original-initials-eur-4-keep-fam.txt  
       ./match.pl -f original-initials-eur-4-keep-fam.txt -g original-initials-eur-4-pca.eigenvec -k 2 -l 2 -v 3 > original-initials-eur-4-keep-pca-fam.txt
    2. Create PC plots using the plot-batch-PCs.R script. See $GITHUB/lib/README-for-plot-batch-PCs for usage instructions. The input is original-initials-eur-4-keep-pca-fam.txt, title is “original EUR, round 3”, and output is original-initials-eur-4-pca.png.
       1. Once done, please post the plots on slack, and copy the plots to the /tsd/p697/data/durable/projects/moba\_qc\_imputation/export/Module\_II\_Plots folder.
       2. Please record the names of the plots in MoBa\_QC\_numbers sheet.

### Remove PC outliers

* + 1. Select the individuals to keep (in this example, we select individuals with PC1 value more than zero; please adjust the awk command as needed). Please reach out to Elizabeth if you have any doubts/questions.  
       awk ‘$3>0 {print $1,$2}’ original-initials-eur-4-pca.eigenvec > original-initials-eur-4-pca-keep.txt
    2. Create PC plots after outlier selection
       1. Create a file containing the individuals kept with founders and non-founder status, using the same threshold conditions as step 22.4.1.  
          awk ‘$3>0 {print $0}’ original-initials-eur-4-keep-pca-fam.txt > original-initials-eur-4-pca-fam-keep.txt
       2. Run the plot-batch-PCs.R script. See $GITHUB/lib/README-for-plot-batch-PCs for usage instructions. The input is original-initials-eur-4-pca-fam-keep.txt, title is “original EUR, round 3”, and output is original-initials-eur-4-pca-keep.png
       3. Once done, please post the plots on slack, and copy the plots to the /tsd/p697/data/durable/projects/moba\_qc\_imputation/export/Module\_II\_Plots folder.
       4. Please record the names of the plots in MoBa\_QC\_numbers sheet.
    3. Create PLINK files without the outliers  
       plink --bfile original-initials-eur-4-keep --keep original-initials-eur-4-pca-keep.txt --make-bed --out original-initials-eur-3-round-selection
    4. Please record the number of individuals kept in MoBa\_QC\_numbers sheet (you can see the number in the log file of the above PLINK command).
    5. If you had outliers, add their IDs to those who were removed previously:  
       ./match.pl -f original-initials-eur-3-round-selection.fam -g original-initials-eur-4-keep.fam -k 2 -l 2 -v 1 | awk ‘$7==”-” {print $1,$2,”PCA-without-1kg-round-3”}’>> original-initials-eur-3-removed-individuals.txt

## Plate effects

If you had outliers in step 22, use the original-initials-eur-3-round-selection bfiles. If you did not have any outliers use the original-initials-eur-4-keep bfiles.

### PCA

* + 1. Prune the data  
       plink --bfile original-initials-eur-3-round-selection --indep-pairwise 3000 1500 0.1 --out original-initials-eur-3-round-indep
    2. Remove long stretches of LD and extract the SNPs
       1. Copy the file containing the list of long LD regions (Build37) from the “resources” folder to your working directory (name of the file is “high-ld.txt”)
       2. Extract pruned SNPs and Remove long stretches of LD  
          plink --bfile original-initials-eur-3-round-selection --extract original-initials-eur-3-round-indep.prune.in --make-set high-ld.txt --write-set --out original-initials-eur-3-round-highld  
          plink --bfile original-initials-eur-3-round-selection --extract original-initials-eur-3-round-indep.prune.in --exclude original-initials-eur-3-round-highld.set --make-bed --out original-initials-eur-3-round-trimmed
    3. Copy the populations.txt file from “resources” folder on DATA/DURABLE to your working directory. populations.txt is a text file containing the population, based on which PLINK will calculate the main PCs, in this case it will be just one word “parent”).
    4. Run the PCA  
       plink --bfile original-initials-eur-3-round-trimmed --pca --within original-initials-eur-4-populations.txt --pca-clusters populations.txt --out original-initials-eur-4-round-pca

### Plot PCs by plate

* + 1. Copy the plate file relevant to your array from “resources” folder to your working directory: (HCE-plates.txt, GSA-plates.txt, or OMNI-plates.txt). These files contain the information on where each individual’s DNA was plated. Throughout this section the HCE-plates.txt file is used as an example. Please make sure to use update the code to match the array used to genotype your batch.
    2. Add plate information to a file with PCA results and make exploratory plots:  
       ./match.pl -f HCE-plates.txt -g original-initials-eur-4-round-pca.eigenvec -k 1 -l 2 -v 3 | awk ‘$23!=”-” {print $0}’ | sort -k23 > original-initials-eur-4-pca-plates.txt  
       awk ‘{print $1,$2,$23}’ original-initials-eur-4-pca-plates.txt > original-initials-eur-4-plate-groups.txt
    3. Create exploratory plots of PCs colored by plate using the plot-PC-by-plate.R script. See $GITHUB/lib/README-for-plot-PC-by-plate for usage instructions. The input is original-initials-eur-4-pca-plates.txt, the title is “original EUR, round 3”, and the output is original-initials-eur-4. Example: Rscript $GITHUB/lib/plot-PC-by-plate.R m24-tz-eur-4-pca-plates.txt “m24 EUR, round3” m24-tz-eur-4
    4. Once done, please post the plots on slack, and copy the plots to the /tsd/p697/data/durable/projects/moba\_qc\_imputation/export/Module\_II\_Plots folder.
    5. Please record the names of the plots in the MoBa\_QC\_numbers sheet.

### ANOVA

* + 1. To perform ANOVA for the first 10PCs and the plates run the anova-for-PC-vs-plates.R script. See $GITHUB/lib/README-for-anova-for-PC-vs-plates for usage instructions. Please specify the input as original-initials-eur-4-pca-plates.txt and the output as original-initials-eur-4-pca-anova-results.txt
    2. Once done, please post the results on slack. You may use the following command to see the results:  
       more original-initials-eur-4-pca-anova-results.txt
    3. Please record the p-values in the MoBa\_QC\_numbers sheet.

### Test for association between the plate and SNPs

* + 1. Run a Cochran-Mantel-Haenszel test for association in founders using sex as the phenotype.  
       plink --bfile original-initials-eur-3-round-selection --filter-founders --chr 1-22 --pheno original-initials-eur-3-round-selection.fam --mpheno 3 --within original-initials-eur-4-plate-groups.txt --mh2 --out original-initials-eur-4-mh-plates
    2. Create QQ plot using the $GITHUB/lib/plot-qqplot.R script for more information see: <https://github.com/norment/moba_qc_imputation/tree/master/lib#plot-qqplotr>. Usage: Rscript plot-qqplot.R inputfile tag pcol outprefix.  
       Please use the following arguments: inputfile: original-initials-eur-4-mh-plates.cmh2; tag: use the “plot-PLINK” file in “resources” folder to assign the harmonized tag to the plots of your batch, followed by the core subpopulation, and QC round; outprefix: original-initials-eur-4-mh-plates-qq-plot.  
       Example: Rscript $GITHUB/lib/plot-qqplot.R original-initials-eur-4-mh-plates.cmh2 “PLOT-PLINK tag, population, round 3” 5 original-initials-eur-4-mh-plates-qq-plot
    3. Once done, please post the plots on slack, and copy the plots to the /tsd/p697/data/durable/projects/moba\_qc\_imputation/export/Module\_II\_Plots folder.
    4. Please record the name of the plot in the MoBa\_QC\_numbers sheet.
    5. Check if there are any significant SNPs at p-value <0.001 threshold  
       sort -k5 -g original-initials-eur-4-mh-plates.cmh2 | grep -v “NA” > original-initials-eur-4-mh2-plates-sorted  
       awk ‘$5<0.001 {print $2}’ original-initials-eur-4-mh2-plates-sorted > original-initials-eur-4-mh2-plates-significant
    6. Record how many SNPs are significant (with p<0.001) and post that number on slack:  
       wc -l original-initials-eur-4-mh2-plates-significant
    7. Remove the SNPs with significant difference between plates
       1. If you had no outliers in step 22:  
          plink --bfile original-initials-eur-4-keep --exclude original-initials-eur-4-mh2-plates-significant --make-bed --out original-initials-eur-4-batch
       2. If you had outliers in step 22:  
          plink --bfile original-initials-eur-3-round-selection --exclude original-initials-eur-4-mh2-plates-significant --make-bed --out original-initials-eur-4-batch
    8. Add the IDs of SNPs removed here to those that were removed previously:  
       awk ‘{print $1,”plate-effect”}’ original-initials-eur-4-mh2-plates-significant >> original-initials-eur-3-bad-snps.txt

### Re-run PCA

* + 1. Prune the data  
       plink --bfile original-initials-eur-4-batch --indep-pairwise 3000 1500 0.1 --out original-initials-eur-4-batch-indep
    2. Remove long stretches of LD and extract the SNPs  
       plink --bfile original-initials-eur-4-batch --extract original-initials-eur-4-batch-indep.prune.in --make-set high-ld.txt --write-set --out original-initials-eur-4-batch-highld  
       plink --bfile original-initials-eur-4-batch --extract original-initials-eur-4-batch-indep.prune.in --exclude original-initials-eur-4-batch-highld.set --make-bed --out original-initials-eur-4-batch-trimmed
    3. Copy the populations.txt file from “resources” folder on DATA/DURABLE to your working directory. populations.txt is a text file containing the population, based on which PLINK will calculate the main PCs, in this case it will be just one word “parent”).
    4. Run the PCA  
       plink --bfile original-initials-eur-4-batch-trimmed --pca --within original-initials-eur-4-populations.txt --pca-clusters populations.txt --out original-initials-eur-4-batch-pca

### Re-run ANOVA

* + 1. Add plate information to a file with PCA results:  
       ./match.pl -f HCE-plates.txt -g original-initials-eur-4-batch-pca.eigenvec -k 1 -l 2 -v 3 | awk ‘$23!=”-” {print $0}’ | sort -k23 > original-initials-eur-4-batch-pca-plates.txt
    2. To perform ANOVA for the first 10PCs and the plates run the anova-for-PC-vs-plates.R script. See $GITHUB/lib/README-for-anova-for-PC-vs-plates for usage instructions. Please specify the input as original-initials-eur-4-batch-pca-plates.txt and the output as original-initials-eur-4-batch-pca-anova-results.txt
    3. Once done, please post the results on slack. You may use the following command to see the results:  
       more original-initials-eur-4-batch-pca-anova-results.txt