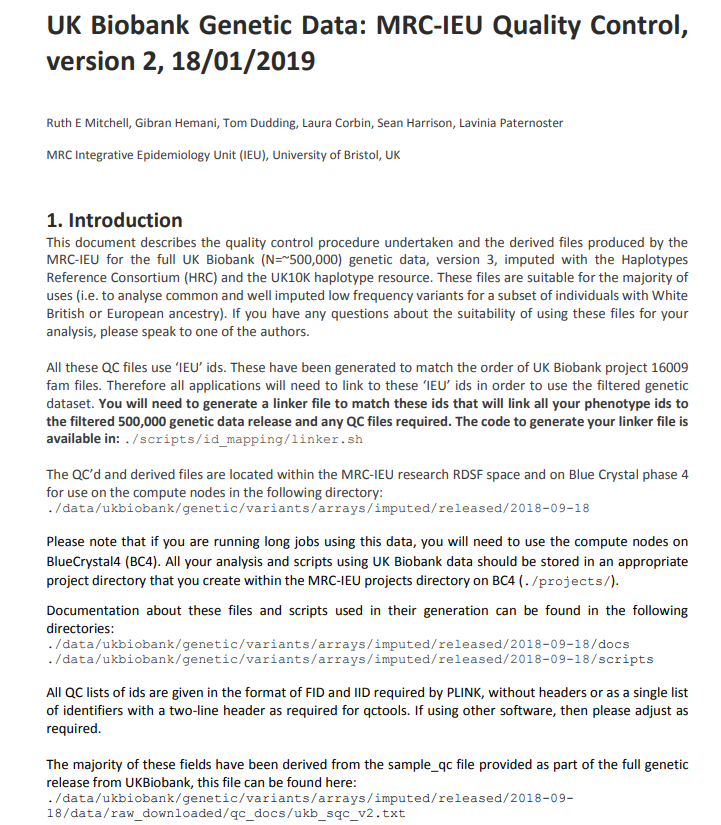
# Using UK Biobank Genetic data

The MRC Integrative Epidemiology Unit (IEU) has stored the UK Biobank genetic data, containing 11,511,739 SNPs from the ~500,000 participants in UK Biobank. This guide is intended to help people find, access and use the data.

## 1.a. Where is the data?

1. Data is stored on the RDSF (Blue Crystal – BC3 or BC4)
   1. /projects/MRC-IEU/research/data/ukbiobank/genetic/variants/arrays/imputed/released/2018-09-18/data
   2. Note that the previous version of UK Biobank genetic data (before January 2019) has been superseded, please use the new version!
2. Using the dosage\_bgen files inside the data folder is recommended, as this data has been filtered
   1. How has it been filtered? See here: https://data.bris.ac.uk/datasets/1ovaau5sxunp2cv8rcy88688v/UK%20Biobank%20Genetic%20Data\_MRC%20IEU%20Quality%20Control%20version%202.pdf



## 1.b. Access to the data

1. Your UK Biobank application needs to have requested access to the genetic data and the genetic fields need to be in your AMS basket
2. Request access to the 500k HRC imputed data from the IEU data management email address
   1. [ieu-datamanagement@bristol.ac.uk](mailto:ieu-datamanagement@bristol.ac.uk)
3. You need a username and password for Blue Crystal:
   1. There’s a form online: <https://www.acrc.bris.ac.uk/login-area/apply.cgi>
   2. You get an email asking to set up a NEW password once you are created as a user
   3. This can be the same password as your normal Bristol user (it automatically is for BC4)
   4. The form asks for your preferred log-in shell – use bash unless you know better

## 1.c. Using Blue Crystal

<https://www.acrc.bris.ac.uk/protected/bc4-docs/>

<https://www.acrc.bris.ac.uk/acrc/pdf/bc-user-guide.pdf> [this is guidance for Blue Crystal 3, but a lot of this is still relevant to BC4)

1. Blue crystal is accessed via a shell program and uses Linux as the operating system
2. Use **WinSCP** to move files across to your user area on Blue Crystal
   1. Instructions on how to login to WinSCP are in top link above
3. Use **Putty** to run shell scripts
   1. type: **qsub** [**filename**] to submit a script to the computer nodes, where [**filename**] is the filename of the shell script, located in the same folder as you are currently in using putty
   2. You can check on the progress of the script using: **qstat -u [USERNAME]**
   3. type **pwd** (print working directory) to check where you are, and **cd** [**new folder path**] to change to a different location
4. You can also type commands straight into **Putty**. This is useful if you want to do something quickly, as it can take a while for a job submitted to BC to run. However, this is not recommended as the login node serves everyone using BC, so if it’s processing for you it will be slower for everyone else. You may eventually get an angry email from HPC.
   1. Rule of thumb: if a job is going to take more than 10-20 minutes, do NOT run it on the login node

## 1.d. What is a shell script?

A shell script (.sh file extension) contains all the instructions for Blue Crystal to run. It needs to be headed by some instructions if you are using Blue Crystal:

|  |
| --- |
| #!/bin/bash  #PBS -l nodes=1:ppn=16  #PBS -l walltime=00:12:00:00  #PBS -N get\_snps  cd $PBS\_O\_WORKDIR |

This tells Blue Crystal that it’s bash script (and the path to the shell on the RDSF), that you want 1 node with 16 processors, that you want the processors for 12 hours, and also to set the directory to be correct (cd = change directory). The user guides for BC explains all this.

Increase the number of nodes, processor and/or runtime if it’s a big job – the program will end at the specified walltime regardless of whether the job has been completed. The rest of the shell script is then a series of commands.

## 1.e. Loading apps, languages, programs

BC has many apps etc. that can be loaded in, e.g. R, Stata, Plink, but these are NOT loaded in by default. Type **module available** to see all of them, and **module load [path]** to load them. **module list** tells you the currently loaded modules.

## 1.f. How is the data presented?

The recommended data (**dosage\_bgen**) is stored in the bgen file format, which comes with associated bgen.bgi and .sample files. This is a UK Biobank specific file format, which neatly stores a huge amount of genetic data in (relatively) small files. The files are split by chromosome and contain enough information to work out for each individual in UK Biobank their dosage of effect allele for each of the 11.5 million SNPs measured. You can learn more here:

<http://www.well.ox.ac.uk/~gav/bgen_format/>

Bgen files can’t be used directly as these are binary files, you need intermediate programs to convert them to useable files. Bgenix can be used to extract particular SNPs, and Plink2 can turn the BGEN files into dosage data (i.e. 0, 1, 2 copies of the effect allele).

As the .bgen files contain all 11.5 million SNPs, it is usual to provide a list of rs IDs so Bgenix can extract only the relevant SNPs. The list can be a simple text list of rs IDs, one per line, without quotes. See the examples for the code to do this.

If you try to extract everything or convert everything into dosage data, you’ll likely run into memory errors.

## 1.g. Creating a linker file for merging with phenotype data

The genetic has different, IEU-specific IDs to each of the phenotype IDs, so a “linker file” is needed to connect the IDs.

This is a fiddly process.

The phenotype IDs are provided in a *.fam* file which is provided UK Biobank.

To generate a linker file, a *.fam* file must first be downloaded using the **ukbgene** executable that came with the PHENOTYPIC data, and a *.ukbkey* file that was emailed to the project PI from UK Biobank. Go to your application on AMS and download the ukbgene utility. You can use this utility to download the fam file - documentation can be found here:

<https://biobank.ndph.ox.ac.uk/showcase/refer.cgi?id=664>)

The standard **ukbgene** executable does NOT work on Blue Crystal (incompatibility with the version of Linux). It will run on epi-Franklin, so if you have access to that skip to the next set of instructions. Otherwise you need to build the ukgene yourself following these instructions. You will need a Blue Crystal account and access to Putty and WinSCP (see **1.b**).

***NOTE: this process doesn’t seem to be working on BLUE CRYSTAL 4, but does work on BLUE CRYSTAL 3. If you run into problems, e.g. “ukbgene: command not found”, then run everything on BLUE CRYSTAL 3.***

1. Download the **ukbgene\_linkset.tar** file from:
   1. <http://biobank.ndph.ox.ac.uk/showcase/refer.cgi?id=665>
2. Copy this file to your Blue Crystal home directory
3. Run the following command in Putty to extract the relevant files:
   1. **tar -xvf "ukbgene\_linkset.tar"**
4. Once complete, type: **md5sum Makefile ukbgene.cpp ugene.h libugene.a**
   1. This will run a check on the downloaded data and produce some codes
   2. Compare these to the codes in the **ukbgene.md5** file (you can open this is WinSCP – refresh WinSCP to see this file)
   3. If they are the same, proceed; if not, re-download the files, and if there are more problems contact IT for help
5. Once checked and the same, type: **make ukbgene**
   1. This will generate an executable ukbgene file, the same as if you used the ukbgene file download directly from UK Biobank
6. Type: **ukbgene**
   1. This should tell you which flags you can use (options), and the date of compilation
   2. If it doesn’t, re-download the files and follow the instructions again

Once you have a working version of the **ukbgene** executable file, you need the **.ukbkey** file. This key file is sent by email for each application and a new <project\_number>.key is sent to the PI on the application at each new release of phenotypic data.

1. Copy the key file across to you Blue Crystal home directory
2. Rename the key file “**.ukbkey**”, i.e. remove any text before the file extension
3. This will make the file invisible, this is ok – it’s the format that the ukgene utility wants it in
4. Run the command: **ukbgene cal -c17 -m**
   1. This will create a **.fam** file – this is the file you need, so copy that back to your working folder
   2. The file will be called something like the following:
      1. **ukb[appnumber]\_cal\_chr17\_v2\_s488292.fam**
5. You can remove all the files you copied across to Blue Crystal now you have the .fam file

The .fam file is NOT the linker file. It is used to *generate* the linker file, using code that is currently in the script folder of the UK Biobank genetic data (**linker.sh**):

**/projects/MRC-IEU/research/data/ukbiobank/genetic/variants/arrays/imputed/released/2018-09-18/scripts/id\_mapping**

The **linker.sh** file can be copied to your home directory, and you will need to read through and change some details (e.g. “appnumber = [something]”). Let Ruth Mitchell know if you have problems running this code. If run correctly, it will produce a linker file called **linker.csv**.

The linker file is simply a list of IEU genotype IDs, and a list of phenotype IDs. The *merge* or *match* commands in R or Stata can be used to merge the genotype IDs with the linker file, then the merge file with phenotypes (or vice versa). Note that all UK Biobank applications will need their own linker file, they cannot be used interchangeably.

## 1.h. Anything else?

There are exclusions you may want to make to the genetic data, e.g. those with a reported sex different to their genetic sex, people with aneuploidy, or ancestry exclusions. Also, it is reasonably standard to only use white British participants, and to exclude related individuals. See here again: <https://data.bris.ac.uk/datasets/1ovaau5sxunp2cv8rcy88688v/UK%20Biobank%20Genetic%20Data_MRC%20IEU%20Quality%20Control%20version%202.pdf>

There are lists of genotypic IDs for excluding participants in:

**/projects/MRC-IEU/research/data/ukbiobank/genetic/variants/arrays/imputed/released/2018-09-18/data/derived**

You can use these lists in R, Stata, Plink or QC Tools to remove IDs. There are two IDs listed for each individual – this is just so Plink/QC Tools can deal with the data – if you are using R/Stata then you just need one column.

If you are using Stata, you can create a .do file that does the same job:

1. Copy the lists of IDs you want to drop from the text files in the above folders to the first column in Excel
2. In cell B1, type the following: **=CONCATENATE("capture drop if id\_gen == ",A1)**
3. Copy this formula into all the cells next to IDs, making sure there are no blank IDs
4. Copy all the resulting cells into a .do file, and save it
5. Load the genetic data you want to remove people from
6. Make sure the ID variable is named “id\_gen”
7. Use the command: **run “[location of do file you just saved]”**
8. This will remove all the IDs you wanted to exclude

# Examples

## Selecting SNPs - Bgenix

This code does the following:

1. Loads the bgen module
2. Cycles through the 23 chromosomes (“for chrom in {1,2…X}; do”)
3. Extracts the SNPs inside a text file called “snp\_list.txt” and creates 23 individual files (“bgenix …”)
4. Combines the 23 files into a single file called “instruments.bgen”, containing only the SNPs specified in “snp\_list.txt” (“cat-bgen…”)
5. Removes the 23 individual files it created (“rm $temp\_geno\_prefix\*”)

|  |
| --- |
| module load apps/bgen-1.0.1  bgen\_pattern=/projects/MRC-IEU/research/data/ukbiobank/genetic/variants/arrays/imputed/released/2018-09-18/data/dosage\_bgen/data.chrCHROM.bgen  bgen\_index\_pattern=/projects/MRC-IEU/research/data/ukbiobank/genetic/variants/arrays/imputed/released/2018-09-18/data/dosage\_bgen/data.chrCHROM.bgen.bgi  snp\_list=snp\_list.txt  temp\_geno\_prefix=temp\_genos  for chrom in {01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,22,X}; do  chrom\_padd=$(printf "%0\*d\n" 2 $chrom)  inbgen=${bgen\_pattern/CHROM/$chrom\_padd}  inbgenidx=${bgen\_index\_pattern/CHROM/$chrom\_padd}  **bgenix -g $inbgen -i $inbgenidx -incl-rsids $snp\_list > $temp\_geno\_prefix.$chrom\_padd.bgen**  done  cmd=""  for chrom in {01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,22,X}; do  cmd="${cmd} ${temp\_geno\_prefix}.${chrom}.bgen"  done  cat-bgen -g ${cmd} -og instruments.bgen  # Remove temp genos  rm $temp\_geno\_prefix\* |

* **bgen\_pattern** and **bgen\_index\_pattern** tell the code where the bgen files are located.
* **snp\_list.txt** can be named anything you like (just change the second line of code). It needs to be a list of rs IDs, one per line, with no quotes.
* **temp\_geno\_prefix** is a placeholder and could be any string.
* The **bgenix** command is in bold; the options are:
  + -g [bgen file]
  + -i [bgen.bgi file]
  + -incl-rsids [snp list file, a text list of rs IDs (no quotes)]
* The “>” tells Linux to output the results to a file, rather than the window. All instances of “$” are local macros/temporary variables/placeholders, e.g. “$chrom” cycles from 1 to 22 (plus X), since it is the loop counter.

## Turning bgen file into raw file – Plink 2.00

This code loads the Plink 2.00 module, then takes the bgen files and converts it to a raw file containing dosages for all participants.

|  |
| --- |
| module add apps/plink/2.00  plink2 --bgen instruments.bgen --hard-call-threshold 0.4999 --export A --out instruments |

The plink options are:

* --bgen [.bgen file from Bgenix]
* --hard-call-threshold [#] #removes dosages that are # far from an integer. 0.4999 keeps pretty much all the results, if you need to, remove the dubious imputes later
* --export [A] #tells plink to export a .raw file
* --out [string] #Names the output file

raw files can be imported into Stata using File->import->Text data (delim), or into R using:

**data = as.data.frame(fread("instruments.raw", header = T, sep = "\t"))**

## Clumping – Plink 1.9

This code loads Plink 1.9 (Plink 2.00 doesn’t work here for some reason), then uses the eur.bed, eur.bim, eur.fam files currently on Scratch on the RDSF “/projects/MRC-IEU/scratch”, but ask [sean.harrison@bristol.ac.uk](mailto:sean.harrison@bristol.ac.uk) for access if you can’t find them and a list of SNPs with P values to clump the SNPs. Look here for more info: <http://zzz.bwh.harvard.edu/plink/clump.shtml>

|  |
| --- |
| module add apps/plink/1.90  plink -bfile eur --clump unclumped\_snps.txt --clump-p1 1 --clump-p2 1 --clump-r2 0.001 --out clumped\_snps  tr -s [:blank:] < clumped\_snps.clumped | cut -f 4 -d " " > clumped\_snps.txt |

The list of SNPs needs to have the headers “SNP” and “P”. The P values are for the association of the SNP with the trait of interest and are used to select the SNPs to preferentially keep. These can be set to a random number if unknown and you aren’t worried about which SNPs to keep. The eur\* files are the Europeans from 1000 genomes, and is used to find blocks to clump.

The plink options are:

* -bfile [eur files]
* --clump [SNP & P text list, no quotes]
* --clump-p1 [1] #This sets the upper limit of P values of index SNPs, set to 1 to keep everything
* --clump-p2 [1] #As above for clumped SNPs
* --clump-r2 [#] #Sets the limit for what is considered a block, 0.001 is fine
* --out [string] #Names the output file

The *tr* command trims the created file of spaces and the *cut* command selects the list of clumped SNPs, outputting a new file called clumped\_snps.txt. This was the quickest way I found of extracting the clumped SNP list. The resulting SNP list can be used to remove unnecessary SNPs from a different dataset in R or Stata.

## Creating a Genetic Risk Score – Plink 2.0

Plink can be used to create Genetic Risk Scores, although the **mrbase\_grs function** can be used to save time and effort. Email [sean.harrison@bristol.ac.uk](mailto:sean.harrison@bristol.ac.uk) to use it – the function downloads SNPs from MR Base, harmonises data, looks for proxies and clumps if necessary, making the whole process easier.

However, you may still wish to use Plink. The code does the following:

1. Loads the Bgenix and Plink 2.0 modules
2. Iterates through the 23 chromosomes
3. Uses Bgenix to extract the relevant SNPs
4. Use plink’s --score option with a text file containing a list of SNPs, their effect alleles, and the beta of the SNP’s association with the trait of interest. This creates 23 sscore files, which can be read into R and merged.
5. Removes temporary bgen files

|  |
| --- |
| module load apps/bgen-1.0.1  module add apps/plink/2.00  bgen\_pattern=/panfs/panasas01/dedicated-mrcieu/research/data/ukbiobank/\_latest/UKBIOBANK\_Array\_Genotypes\_500k\_HRC\_Imputation/data/dosage\_bgen/data.chrCHROM.bgen  bgen\_index\_pattern=/panfs/panasas01/dedicated-mrcieu/research/data/ukbiobank/\_latest/UKBIOBANK\_Array\_Genotypes\_500k\_HRC\_Imputation/data/dosage\_bgen/data.chrCHROM.bgen.bgi  snp\_list=snp\_list.txt  snp\_list\_alleles = snp\_list\_alleles.txt  temp\_geno\_prefix=temp\_genos  for chrom in {01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,22,X}; do  chrom\_padd=$(printf "%0\*d\n" 2 $chrom)  inbgen=${bgen\_pattern/CHROM/$chrom\_padd}  inbgenidx=${bgen\_index\_pattern/CHROM/$chrom\_padd}  bgenix -g $inbgen -i $inbgenidx -incl-rsids $snp\_list > $temp\_geno\_prefix.$chrom\_padd.bgen  plink --bgen $temp\_geno\_prefix.$chrom\_padd.bgen --score $snp\_list\_alleles --out $temp\_geno\_prefix.$chrom\_padd  rm ${temp\_geno\_prefix}\*.bgen  done |

The bgenix options are as above, but the plink options now include:

* --score [SNP list file] #The snp\_list\_alleles.txt file must have in rs IDs in column 1, effect alleles in column 2, and betas in column 3, with no quotes and no header

The score from plink is an average across all alleles, so needs to be manipulated within each file to get a more interpretable GRS. Example **R code** for importing the score and modifying the GRS is provided below. The modification is simply that average needs to be multiplied by the number of non-missing alleles to get the total number of effect alleles multiplied by their respective betas. There are options to produce a sum, rather than an average, check online for details.

|  |
| --- |
| **#Note, this is R code**  grs = data.frame()  temp = read.delim("temp\_genos.01.sscore",sep=" ")  grs = data.frame(id=temp[,c('IID')])  grs$grs = temp[,3]\*temp[,c('SCORE1\_AVG')]  for(i in 2:23) {  if(i < 10){  i = paste('0',i,sep='')  }  if(i == 23) {  i = 'X'  }  temp = read.delim(paste("temp\_genos.",i,".sscore",sep=""),sep=" ")  merge = data.frame(id=temp[,c('IID')])  merge$grs\_temp = temp[,3]\*temp[,c('SCORE1\_AVG')]  grs= merge(grs, merge, by.x='id', by.y='id')  grs$grs = grs$grs+grs$grs\_temp  grs$grs\_temp = NULL  }  write.csv(grs,"grs.csv",row.names=FALSE) |

# Notes

This guide was written because I found it very difficult to grab all the relevant bits of information needed to use the UK Biobank genetic data. It was only by talking with Ruth Mitchell, Tom Richardson, Teri-Louise North and Neil Davies that I managed to do any of this.

Hopefully with this guide, the process will be much easier in future. Any suggestions or questions, please email me, but bear in mind I probably won’t know the answer!

--Update 05/04/2019

I’ve updated this manual so it is compatible with using the new UK Biobank genetic data. I *think* I’ve caught everything, let me know if something looks wrong though.

**--Sean Harrison, 05/04/2019,** [**sean.harrison@bristol.ac.uk**](mailto:sean.harrison@bristol.ac.uk)