# Downloading and Using UK Biobank data

## Contents

* Phenotypic data
  + Section 1: Downloading phenotypic data
  + Section 2: Extracting phenotypic data
  + Section 3: Using phenotypic data
  + Code
* Hospital Episode Statistics (HES) data
  + Section 1: Downloading HES data
  + Section 2: Using HES data
* Primary care data
  + Section 1: Downloading primary care data
  + Section 2: Using primary care data
* Genetic data (Note: this is a copy of *Using UK Biobank genetic data v.02*)
  + 1.a. Where is the data?
  + 1.b. Access to the data
  + 1.c. Using Blue Crystal
  + 1.d. What is a shell script?
  + 1.e. Loading apps, languages, programs
  + 1.f. How is the data presented?
  + 1.g. Creating a linker file for merging with phenotype data
  + 1.h. Anything else?
  + Examples
    - Selecting SNPs – Bgenix
    - Turning bgen file into raw file – Plink 2.00
    - Clumping – Plink 1.9
    - Creating a Genetic Risk Score – Plink 2.00
* Creating a Genetic Risk Score Using the mrbase\_grs R Function (Note: this is a copy of *mrbase\_grs documentation v.02*)
  + 1. Options
  + 2. Additional Files
  + 3. Code Procedures
  + 4. Running the Program
  + 5. Additional Notes
  + 6. Examples
  + 7. Statement for Papers

**I have recorded a series of videos going through the downloading, extracting and using of the: 1) phenotypic data, 2) HES data, and 3) primary care data. I have also recorded videos showing how to 1) create a linker file, 2) select SNPs with Bgenix, 3) turn a BGEN file to a .raw file, 4) clump SNPs with Plink, 5) create a genetic risk score with plink, and 6) use the mrbase.grs function. These videos are available here: M:\projects\ieu2\p6\006\working\data\Guides to using UK Biobank\Videos**

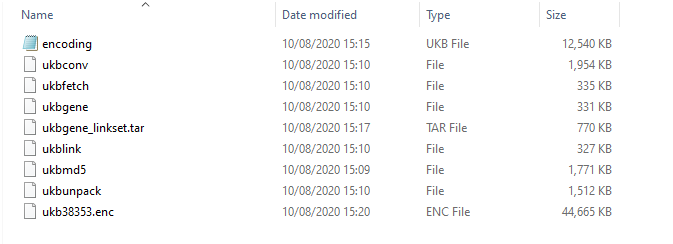
## Phenotypic Data

### Section 1: Downloading phenotypic data

1. Log in to the UK Biobank AMS system: <https://bbams.ndph.ox.ac.uk/ams/researcher_home.jsp>
2. Go to “**Projects**”
3. Select the relevant project, and click “**View/Update**”
4. This will list off all the application details for the project. Select the “**Data**” tab on the right
5. Press the “**Go to Showcase to refresh or download the data**” button, which may take a few seconds to load
6. There’s a link to the “**using UKB data**” guide on this page: <https://biobank.ndph.ox.ac.uk/showcase/exinfo.cgi?src=accessing_data_guide>
7. Which contains a link to the “**Accessing data guide**”: <https://biobank.ndph.ox.ac.uk/~bbdatan/Accessing_UKB_data_v2.1.pdf>
8. Both webpages are useful, read them, especially section 2 of the “**Accessing data guide**”, the section called “**Downloading a main dataset**”
9. When the data page has loaded, you’ll be able to download (depending on your application) “**File Handlers**”, “**Miscellaneous Utility**”, “**Datasets**”, and have access the to “**Data Portal**”
   1. **File Handlers** are used to manipulate the datasets, and are essential
   2. **Miscellaneous Utility** is the encoding.ukb file, which is also essential
   3. **Datasets** are, unsurprisingly, your phenotypic datasets – if you don’t see this, I would guess you don’t have access to the data: you need to be PI or have delegate access
   4. **Data Portal** is a link to the record repository, which is an SQL database used when downloading HES data
10. Download all file handlers (the linux versions) and the miscellaneous utility
    1. Note: Everything apart from the file encoding,ukb file can be clicked on to save, but for the encoding.ukb file, you need to right click and click “Save link as…”
    2. You don’t need all file handlers, but it’s as well to be prepared
11. Now download the phenotype dataset of your choice
    1. You’ll need an MD5 checksum that was sent to you in an email from UK Biobank, and is specific to the *run ID*, not just the application ID. Hopefully you haven’t lost this.
    2. Hint: It’ll be in an email entitled **UK Biobank Application XXXXX, data now available for Basket XXXXXXX**, sent when the basket was made available for download.
    3. The phenotype file will be named ukbXXXXX.enc, where XXXXX is the run ID number.
    4. Note: the baskets expire over time and aren’t downloadable. I don’t know how long it takes for the baskets to expire, but it’s probably 1 year.
12. We’ll deal with HES and primary care data later.

### Section 2: Extracting phenotypic data

1. You should now have a list of files that looks something like this:



1. Copy these files across to blue crystal 3 using WinSCP
2. Open BC3 on Putty
3. Navigate to the folder in which you stored the files
4. Give the files permission to run by typing: **chmod +x ukb\***
5. Validate the download:
   1. Type **ukbmd5 ukbXXXXX.enc**, replacing XXXXX with your run ID
   2. This will spit out an MD5 checksum, which **NEEDS** to be identical to the MD5 checksum you entered in order to download the phenotypic dataset (you know, the one in the email you hopefully didn’t lose)
   3. If it isn’t identical, re-download the dataset. I had this happen once, so it isn’t impossible and is necessary.
6. Decrypt the dataset:
   1. First, find the *keyvalue* password – this should have been attached to the email you definitely didn’t lose – and copy it (it’s the 64 characters below your application ID in the attached file)
   2. Type **ukbunpack ukbXXXXX.enc *keyvalue***, where XXXXX is run ID, and *keyvalue* is the 64 character password you just copied
   3. This… May take a while.
7. Convert the dataset to something usable:
   1. The decrypted unpacked dataset is unusable, so convert it to something usable
   2. Options are csv, txt, r, sas or stata – we’ll do Stata here
   3. Type: **ukbconv ukbXXXXX.enc\_ukb option**, where option is one of the above file formats
   4. This will take longer than unpacking. Maybe go for a walk or something if you have hundreds of thousands of variables.
   5. You can do this repeatedly with different options, so can create Stata and R versions if you like.
8. For Stata, this conversion creates a .raw file (like a CSV without the commas), a .do file, and a .dct (dictionary) file. The .raw file doesn’t have ANY coding, so is functionally useless. The .do file is created to label and format the data, along with the dictionary file – it’s really quite clever.
9. BUT there’s a snag: if your dataset is large (my biggest one was 48 GB), the you can’t load it into Stata all at once. It just crashes. Your maximum usable memory is likely slightly less than the size of your RAM, and once coded, .dta files can be larger than the .raw equivalent (although they can be smaller, go figure). As such, you have two options, depending on how large the .raw file is and how much RAM you have:
   1. Option 1: if the .raw file is small enough, you can copy the .raw, .dct and .do files to Windows and run the do file there. You’ll need to change the *change directory* line, and maybe add in a save line right at the end, but you’re basically good to go. You could also do this in blue crystal 3, but you would *have* to add the save line at the end, and it would only work if you have fewer than 2,048 variables, as the blue crystal 3 version of Stata has a low variable cap.
   2. Option 2: here’s where things get fun. We’re going to use Linux commands to separate the .raw file (which has close to 510,000 participants) into 51 files of 10,000 participants each. This will reduce the file size, allowing us to run the .do file on each of the 51 files. We still won’t be able to join up the Stata file outputs, since the file will be too large. Instead, we’ll save each of the 51 Stata output files, and use some extra code to select from each of the 51 files the variables we want for any specific project, then join those files together.
10. If you could use Option 1, great! You’re good to go with the phenotypic data.
11. If not, read on to section 3.

### Section 3: Using phenotypic data

1. From the folder with the .raw file in Blue Crystal 3, use the **sed** command to select out the first 10,000 rows of data from the .raw file: **sed -n 1,10000p ukbXXXXX.raw > 1.raw**
2. Repeat that 50 more times, to give 51 datasets, called 1.raw, 2.raw … 51.raw
   1. I’ve put the code at the bottom of the page, just replace all instances of XXXXX with your run ID, and copy it all into putty.
   2. It’ll run the first 50 lines sequentially and automatically, but then you’ll need to press enter for the last line to run.
3. Copy the 51 .raw files into a folder called **x** in your phenotypic folder. Also create a folder called **y**.
   1. Because we keep requesting more data, in the phenotypic folder, we have different folders for different run IDs, and within each, folders called **x** and **y**.
   2. Note: the names are arbitrary, but all my code uses these names. If you call them something else, just remember to edit my code to refer to the new names.
4. Copy the .dct and .do files over to windows too – I put mine in a folder called **Processed files**
5. Edit the .do file – you need to ***remove*** three lines of code, as we’re going to load in each of the 51 .raw files separately, label them, then save them individually:
   1. **clear all**
   2. **cd “…”**
   3. **infile using “ukbXXXXX.dct”, using(“ukbXXXXX.raw”)**
   4. Save the .do file, with the new name **biobank\_program.do**
6. Create a new do file, and copy across the relevant code at the bottom of the page, then run it
   1. This will take each of the 51 datasets, label them, and save them in the **y** folder as processed .dta files
7. [Optional step]: I found it helpful to summarise all the variables in the first dataset (**y1**) and create a spreadsheet showing:
   1. UK Biobank variable ID number, it’s label, in how many time-points the data was collected, how many indices of the variable there are at baseline (i.e. how many repeats of the variable there, e.g. there are 40 principal components), the type of data (numeric, labelled numeric, string), some example text, how many unique values there are, total number of observations and mean/median/IQR/SD/min/max if applicable
   2. I then used this spreadsheet to specify which variables I wanted, which, with some additional code, gave me a snippet of Stata code that slots into a section of code that extracts the relevant variables from the 51 datasets, and combines them into a single usable dataset
   3. The code to generate the spreadsheet is at the bottom of the page, you can add it to the end of the *section 3, part 6* code
   4. Once the spreadsheet is created, add the following to the first row in each column Q: **Keep?**, R: **All baseline measures?**, S: **All timepoints?**, U: **code**, V: **Stata code to copy**
   5. I added colour and borders columns Q-S to mark them out, and made cells V1 and V2 coloured
   6. Add the following code to cell U2, then copy it down to all cells in column U: **=IF(Q2=1,IF(S2=1,CONCATENATE(LEFT(A2,LEN(A2)-3),"\* "),IF(R2=1,CONCATENATE(LEFT(A2,LEN(A2)-1),"\* "),CONCATENATE(A2," "))),"")**
   7. Add the following code to cell V2: **=CONCATENATE("use ",TEXTJOIN(" ",TRUE,U2:U3000)," using ",CHAR(34),"$cd\_phenotype\_data\y\y`i'.dta",CHAR(34))**
   8. Now you can place a **1** in columns Q-S to indicate you either a) want the first instance of the variable at baseline (Q), b) want all instances of the variable at baseline (R), or c) want all instance of the variable at all time-points (S)
   9. Variable names will appear in column U after you type a **1** into either column Q, R or S: how much of the variable name appears depends on which column you put the **1** into.
   10. Note: Variable names in UK Biobank have the following format: **[a]\_[b]\_[c]\_[d]**, where **[a]** is a letter, indicating a normal variable (n), or a time variable (ts), or something else, **[b]** is the variable ID (you can search for this in the UK Biobank data showcase to quickly go to the right variable), **[c]** is the instance number, where 0 is baseline, 1 is the first follow-up, and 2 is the second follow-up, and **[d]** is the index within each instance, usually starting at 0
   11. A line of code is generated in cell V2, starting with **use**, followed by all the variables you have requested, and ending with **using “$cd\_phenotype\_data\y\y`I’.dta”** – as part of the next bit of code, you’ll need to specify a global macro called **cd\_phenotype\_data**, which details the folder where the **y** folder is stored
8. Whether you complete step 7 or not, you can use the code below to use just the variables you want from the 51 **y** datasets, save them in the **z** folder, and combine them all into a single usable dataset
9. And that’s it: you should now have a usable dataset to work from!

### Code for Section 3

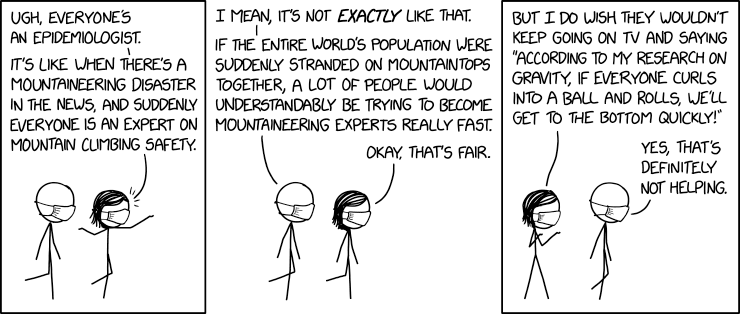
|  |
| --- |
| ***#Section 3, part 3***  #Create 51 datasets from main file for loading into Stata (10,000 participants each)  sed -n 1,10000p ukbXXXXX.raw > 1.raw  sed -n 10001,20000p ukbXXXXX.raw > 2.raw  sed -n 20001,30000p ukbXXXXX.raw > 3.raw  sed -n 30001,40000p ukbXXXXX.raw > 4.raw  sed -n 40001,50000p ukbXXXXX.raw > 5.raw  sed -n 50001,60000p ukbXXXXX.raw > 6.raw  sed -n 60001,70000p ukbXXXXX.raw > 7.raw  sed -n 70001,80000p ukbXXXXX.raw > 8.raw  sed -n 80001,90000p ukbXXXXX.raw > 9.raw  sed -n 90001,100000p ukbXXXXX.raw > 10.raw  sed -n 100001,110000p ukbXXXXX.raw > 11.raw  sed -n 110001,120000p ukbXXXXX.raw > 12.raw  sed -n 120001,130000p ukbXXXXX.raw > 13.raw  sed -n 130001,140000p ukbXXXXX.raw > 14.raw  sed -n 140001,150000p ukbXXXXX.raw > 15.raw  sed -n 150001,160000p ukbXXXXX.raw > 16.raw  sed -n 160001,170000p ukbXXXXX.raw > 17.raw  sed -n 170001,180000p ukbXXXXX.raw > 18.raw  sed -n 180001,190000p ukbXXXXX.raw > 19.raw  sed -n 190001,200000p ukbXXXXX.raw > 20.raw  sed -n 200001,210000p ukbXXXXX.raw > 21.raw  sed -n 210001,220000p ukbXXXXX.raw > 22.raw  sed -n 220001,230000p ukbXXXXX.raw > 23.raw  sed -n 230001,240000p ukbXXXXX.raw > 24.raw  sed -n 240001,250000p ukbXXXXX.raw > 25.raw  sed -n 250001,260000p ukbXXXXX.raw > 26.raw  sed -n 260001,270000p ukbXXXXX.raw > 27.raw  sed -n 270001,280000p ukbXXXXX.raw > 28.raw  sed -n 280001,290000p ukbXXXXX.raw > 29.raw  sed -n 290001,300000p ukbXXXXX.raw > 30.raw  sed -n 300001,310000p ukbXXXXX.raw > 31.raw  sed -n 310001,320000p ukbXXXXX.raw > 32.raw  sed -n 320001,330000p ukbXXXXX.raw > 33.raw  sed -n 330001,340000p ukbXXXXX.raw > 34.raw  sed -n 340001,350000p ukbXXXXX.raw > 35.raw  sed -n 350001,360000p ukbXXXXX.raw > 36.raw  sed -n 360001,370000p ukbXXXXX.raw > 37.raw  sed -n 370001,380000p ukbXXXXX.raw > 38.raw  sed -n 380001,390000p ukbXXXXX.raw > 39.raw  sed -n 390001,400000p ukbXXXXX.raw > 40.raw  sed -n 400001,410000p ukbXXXXX.raw > 41.raw  sed -n 410001,420000p ukbXXXXX.raw > 42.raw  sed -n 420001,430000p ukbXXXXX.raw > 43.raw  sed -n 430001,440000p ukbXXXXX.raw > 44.raw  sed -n 440001,450000p ukbXXXXX.raw > 45.raw  sed -n 450001,460000p ukbXXXXX.raw > 46.raw  sed -n 460001,470000p ukbXXXXX.raw > 47.raw  sed -n 470001,480000p ukbXXXXX.raw > 48.raw  sed -n 480001,490000p ukbXXXXX.raw > 49.raw  sed -n 490001,500000p ukbXXXXX.raw > 50.raw  sed -n 500001,510000p ukbXXXXX.raw > 51.raw |

|  |
| --- |
| ***\*Section 3, part 6***  cd [folder location]  \*Code to generate y datasets  \*i.e. go from .raw data to nice labelled Stata datasets  forvalues i = 1/51 {  clear all  set maxvar 15000    local x1 = (`i'-1)\*10000+1  local x2 = `i'\*10000  dis "Loading observations: `x1' to `x2'"  infile using "Processed files\ukb38353.dct", using("x\\`i'.raw")  qui do "biobank\_program.do"    save "y\y`i'.dta", replace  } |

|  |
| --- |
| ***\*Section 3, part 7c***  \*Code to summarize all variables  use "y\y1.dta", clear  \*Reduce down to first instance in the first time-point  gen variable = "n\_eid" in 1  local varlabel: var label n\_eid  gen label = "`varlabel'" in 1  gen instances = 1 in 1  gen indices = 1 in 1  local i = 2  foreach var of varlist \_all {  if substr("`var'",-4,4) == "\_0\_0" {  replace variable = "`var'" in `i'  local varlabel: var label `var'  replace label = "`varlabel'" in `i'    local l = length("`var'") - 4  local var2 = substr("`var'",1,`l')    \*Indices  local var3 = "`var2'\_0\_\*"  local x = 0  foreach var of varlist `var3' {  local x = `x'+1  }  replace indices = `x' in `i'    \*Instances  local var4 = "`var2'\_\*\_0"  local x = 0  foreach var of varlist `var4' {  local x = `x'+1  }  replace instances = `x' in `i'  local i = `i'+1  }  }  \*Also want to keep the variables without a 0 index  local vars\_to\_add = ""  foreach var of varlist \_all {  if substr("`var'",-4,4) != "\_0\_0" & substr("`var'",-2,1) == "\_" {  local l = length("`var'") - 3  local var2 = substr("`var'",1,`l')    qui count if strpos(variable,"`var2'") > 0  if r(N) == 0 {  local vars\_to\_add = "`vars\_to\_add' `var'"    replace variable = "`var'" in `i'  local varlabel: var label `var'  replace label = "`varlabel'" in `i'    local i = `i'+1  }  }  else if substr("`var'",-4,4) != "\_0\_0" & substr("`var'",-2,1) != "\_" {  local l = length("`var'") - 4  local var2 = substr("`var'",1,`l')    qui count if strpos(variable,"`var2'") > 0  if r(N) == 0 {  local vars\_to\_add = "`vars\_to\_add' `var'"    replace variable = "`var'" in `i'  local varlabel: var label `var'  replace label = "`varlabel'" in `i'    local i = `i'+1  }  }  }  keep n\_eid \*\_0\_0 variable-indices `vars\_to\_add'  \*Summarise all remaining variables  qui {  gen type = ""  gen example\_text = ""  gen unique\_values = .  gen N = .  gen mean = .  gen p25 = .  gen median = .  gen p75 = .  gen sd = .  gen max = .  gen min = .  local i = 1    qui count if variable != ""  local max = r(N)    forvalues i = 1 / `max' {  local var = variable[`i']  capture confirm string variable `var'  if !\_rc {  \*String variable  replace type = "String" in `i'  \*Check for observations  preserve  keep if `var' != ""  if c(N) > 0 {  local continue = 1  }  else {  local continue = 0  }  restore  if `continue' == 1 {  preserve  keep `var'  drop if `var' == ""  local N = c(N)  bysort `var': egen count = count(`var')  duplicates drop `var', force  sort `var'  local max = c(N)  if `max' >= 3 {  local max = 3  }  forvalues k = 1/`max' {  if `k' == 1 {  local a = `var'[`k']  local a1 = count[`k']  local x = "`a' (`a1')"  }  else {  local a = `var'[`k']  local a1 = count[`k']  local x = "`x', `a' (`a1')"  }  }  local unique = c(N)  restore  replace example\_text = "`x'" in `i'  replace unique\_values = `unique' in `i'  replace N = `N' in `i'  }  else {  replace N = 0 in `i'  }  }  else {  \*Numeric variable  if (!missing(`"`: value label `var''"')) {  \*Numeric variable, labelled - treat as string  replace type = "Numeric (Labelled)" in `i'  \*Check for observations  preserve  keep if `var' != .  if c(N) > 0 {  local continue = 1  }  else {  local continue = 0  }  restore  if `continue' == 1 {  preserve  keep `var'  sdecode `var', gen(temp)  drop `var'  rename temp `var'  drop if `var' == ""  local N = c(N)  bysort `var': egen count = count(`var')  duplicates drop `var', force  sort `var'  local max = c(N)  if `max' >= 3 {  local max = 3  }  forvalues k = 1/`max' {  if `k' == 1 {  local a = `var'[`k']  local a1 = count[`k']  local x = "`a' (`a1')"  }  else {  local a = `var'[`k']  local a1 = count[`k']  local x = "`x', `a' (`a1')"  }  }  local unique = c(N)  restore  replace example\_text = "`x'" in `i'  replace unique\_values = `unique' in `i'  replace N = `N' in `i'  }  else {  replace N = 0 in `i'  }  }  else {  \*Numeric variable, no label  replace type = "Numeric" in `i'  \*Check for observations  preserve  keep if `var' != .  if c(N) > 0 {  local continue = 1  }  else {  local continue = 0  }  restore  if `continue' == 1 {  sum `var', d  replace N = r(N) in `i'  replace mean = r(mean) in `i'  replace p25 = r(p25) in `i'  replace median = r(p50) in `i'  replace p75 = r(p75) in `i'  replace sd = r(sd) in `i'  replace max = r(max) in `i'  replace min = r(min) in `i'  preserve  keep `var'  drop if `var' == .  bysort `var': egen count = count(`var')  duplicates drop `var', force  sort `var'  local max = c(N)  if `max' >= 3 {  local max = 3  }  forvalues k = 1/`max' {  if `k' == 1 {  local a = `var'[`k']  local a1 = count[`k']  local x = "`a' (`a1')"  }  else {  local a = `var'[`k']  local a1 = count[`k']  local x = "`x', `a' (`a1')"  }  }  local unique = c(N)  restore  replace example\_text = "`x'" in `i'  replace unique\_values = `unique' in `i'    }  else {  replace N = 0 in `i'  }  }  }  local i = `i'+1  }  }  keep variable - min  drop if variable == ""  drop if variable == "variable" | variable == "instances" | variable == "indices"  gen num = substr(variable,3,.)  replace num = subinstr(num,"\_","",.)  replace num = "0" if num == "eid"  destring num, replace  sort num  drop num  egen x = ends(variable), punct("\_") tail  egen y = ends(x), punct("\_")  destring y, gen(z) force  replace z = 0 if y == "eid"  sort z  drop if z == .  drop x y z  save "variable\_summary.dta", replace  export delim “variable\_summary.csv”, replace |

|  |
| --- |
| ***\*Section 3, part 8***  \*Create a dataset using only the variables you want  cd [SOMEWHERE]  \*Load in phenotypes  forvalues i = 1/51 {  clear all  set maxvar 15000    \*Excel code from "Phenotypes.xlsx" goes here, e.g.  use n\_eid s\_41202\_\* s\_41203\_\* s\_41204\_\* s\_41205\_\* using "$cd\_phenotype\_data\y\y`i'.dta"    save "z`i'.dta", replace  }  use "z1.dta", clear  erase "z1.dta"  forvalues i = 2/51 {  append using "z`i'.dta"  erase "z`i'.dta"  }  compress  save [SOMEWHERE], replace |

*No one has ever explicitly said I shouldn’t separate the sections of a manual on downloading and using data from a large national population study with XKCD comics, so…*



Alt text: If enough people uphill decide to try the rolling strategy, they can make the decision for you.

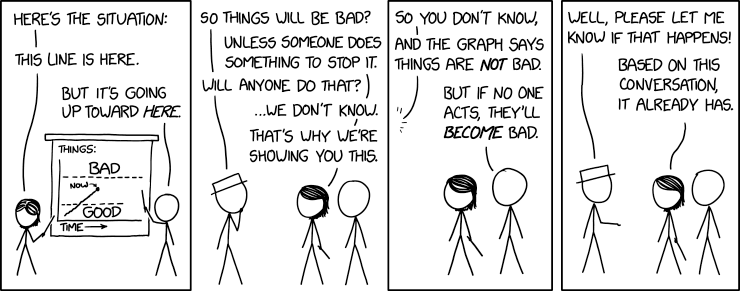
URL: <https://xkcd.com/2300/>

## Hospital Episode Statistic (HES) Data

### Section 1: Downloading HES data

1. Log in to the UK Biobank AMS system: <https://bbams.ndph.ox.ac.uk/ams/researcher_home.jsp>
2. Go to “**Projects**”
3. Select the relevant project, and click “**View/Update**”
4. This will list off all the application details for the project. Select the “**Data**” tab on the right
5. Press the “**Go to Showcase to refresh or download the data**” button, which may take a few seconds to load
6. Go to the **“Data Portal”** tab and click “**Connect”**
7. The data dictionary for the HES data is available here: <http://biobank.ctsu.ox.ac.uk/crystal/refer.cgi?id=141140>
   1. This contains a list of all the tables available in the HES data
8. Navigate to **Table Download**, and fetch the tables: **hesin**, **hesin\_diag10**, and **hesin\_OPER**
   1. Alternatively, if you only want some of the data, you can create an SQL query that returns just the data you want (if you don’t know SQL, it may be quicker to download everything and use Stata or R to grab only the data you want)
9. Once downloaded, the 3 tables need to be modified and joined
   1. I do this in Stata, but you could use R if you like
   2. Code is below

|  |
| --- |
| \*The HESIN\_DIAG file needs some modifications  import delim "HESIN\_DIAG.txt", clear  sort eid ins\_index arr\_index  drop level \*\_nb  rename diag\_icd9 diag\_icd9\_  rename diag\_icd10 diag\_icd10\_  \*Reshape to wide, so we have 1 row per episode  reshape wide diag\_icd9\_ diag\_icd10\_, i(eid ins\_index) j(arr\_index)  save "ukb\_hesin\_diag\_clean.dta", replace  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*The HESIN\_OPER file needs some modications  import delim "HESIN\_OPER.txt", clear  sort eid ins\_index arr\_index  drop level \*\_nb  rename oper3 oper3\_  rename oper4 oper4\_  bysort eid ins\_index: egen x = min(posopdur)  replace posopdur = x if arr\_index != 0  drop x  bysort eid ins\_index: egen x = min(preopdur)  replace preopdur = x if arr\_index != 0  drop x  drop opdate  \*Reshape to wide, so we have 1 row per episode  reshape wide oper3\_ oper4\_, i(eid ins\_index) j(arr\_index)  save "ukb\_hesin\_OPER\_clean.dta", replace  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*Combine HESIN, HESIN\_DIAG and HESIN\_OPER  import delim "HESIN.txt", clear  merge 1:1 eid ins\_index using "ukb\_hesin\_diag\_clean.dta", nogen  merge 1:1 eid ins\_index using "ukb\_hesin\_OPER\_clean.dta", nogen  \*Format dates  gen date\_epistart=date(epistart, "YMD", 2050), a(epistart)  format date\_epistart %td  drop epistart  rename date\_epistart epistart  gen date\_epiend=date(epiend, "YMD", 2050), a(epiend)  format date\_epiend %td  drop epiend  rename date\_epiend epiend  sort eid ins\_index  save "hes.dta", replace |



Alt text: I actually came in in the middle so I don't know which topic we're briefing on; the same slides work for like half of them.

URL: <https://xkcd.com/2278/>

## Primary Care Data

### Section 1: Downloading primary care data

1. Log in to the UK Biobank AMS system: <https://bbams.ndph.ox.ac.uk/ams/researcher_home.jsp>
2. Go to “**Projects**”
3. Select the relevant project, and click “**View/Update**”
4. This will list off all the application details for the project. Select the “**Data**” tab on the right
5. Press the “**Go to Showcase to refresh or download the data**” button, which may take a few seconds to load
6. Go to the **“Data Portal”** tab and click “**Connect”**
7. The PDF explaining primary care is here: <http://biobank.ndph.ox.ac.uk/showcase/showcase/docs/primary_care_data.pdf>
   1. This contains a list of all the tables available in the primary care data (section 6)
8. Navigate to **Table Download**, and fetch the tables: **gp\_registrations**, **gp\_clinical**, and **gp\_scripts**
   1. These will take *ages* to download
9. Once downloaded, the files will work without further manipulation
   1. However, they are very big, so it may make sense to separate the clinical events and prescriptions into blocks of 1,000,000 (as with the phenotypes above), and work with them in blocks rather than as a whole
   2. Alternatively, since the clinical events and prescriptions cover the early 1990s to 2017, restricting the entries to those within the time-period of interest may reduce the file size down to something manageable
10. [Optional] If you want to separate the clinical and script files into something more manageable, copy them across to Blue Crystal using WinSCP, then in Putty run the code below
    1. The two shell scripts (the ones with **sed** at the start of each line) chunk the files into rows of 1,000,000 lines each, creating numbered files
    2. Because the numbered files are the same, copy across the chunked files from one script before starting the next script
    3. I created folders called **clinical** and **scripts** inside a **primary care** folder, then copied the numbered files into folders called **x**
    4. As in the phenotype data, I used code to import all the numbered files to Stata, formatted them, then saved them in a **y** folder
    5. I also created a **z** folder, which I used to save copies of the numbered files that were restricted to just the dates I wanted, appending them together at the end to create complete clinical and script datasets to use
    6. All code is below

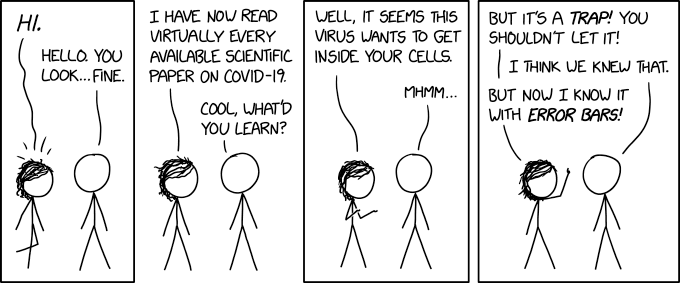
### Primary Care Code

|  |
| --- |
| #!/bin/bash  #PBS -l nodes=1:ppn=16  #PBS -l walltime=00:12:00:00  #PBS -N gp\_scripts  cd $PBS\_O\_WORKDIR  #Shell script for Blue Crystal  #Create 58 datasets from scripts.txt file for loading into Stata (1,000,000 rows each)  sed -n 1,1000000p scripts.txt > 1.txt  sed -n 1000001,2000000p scripts.txt > 2.txt  sed -n 2000001,3000000p scripts.txt > 3.txt  sed -n 3000001,4000000p scripts.txt > 4.txt  sed -n 4000001,5000000p scripts.txt > 5.txt  sed -n 5000001,6000000p scripts.txt > 6.txt  sed -n 6000001,7000000p scripts.txt > 7.txt  sed -n 7000001,8000000p scripts.txt > 8.txt  sed -n 8000001,9000000p scripts.txt > 9.txt  sed -n 9000001,10000000p scripts.txt > 10.txt  sed -n 10000001,11000000p scripts.txt > 11.txt  sed -n 11000001,12000000p scripts.txt > 12.txt  sed -n 12000001,13000000p scripts.txt > 13.txt  sed -n 13000001,14000000p scripts.txt > 14.txt  sed -n 14000001,15000000p scripts.txt > 15.txt  sed -n 15000001,16000000p scripts.txt > 16.txt  sed -n 16000001,17000000p scripts.txt > 17.txt  sed -n 17000001,18000000p scripts.txt > 18.txt  sed -n 18000001,19000000p scripts.txt > 19.txt  sed -n 19000001,20000000p scripts.txt > 20.txt  sed -n 20000001,21000000p scripts.txt > 21.txt  sed -n 21000001,22000000p scripts.txt > 22.txt  sed -n 22000001,23000000p scripts.txt > 23.txt  sed -n 23000001,24000000p scripts.txt > 24.txt  sed -n 24000001,25000000p scripts.txt > 25.txt  sed -n 25000001,26000000p scripts.txt > 26.txt  sed -n 26000001,27000000p scripts.txt > 27.txt  sed -n 27000001,28000000p scripts.txt > 28.txt  sed -n 28000001,29000000p scripts.txt > 29.txt  sed -n 29000001,30000000p scripts.txt > 30.txt  sed -n 30000001,31000000p scripts.txt > 31.txt  sed -n 31000001,32000000p scripts.txt > 32.txt  sed -n 32000001,33000000p scripts.txt > 33.txt  sed -n 33000001,34000000p scripts.txt > 34.txt  sed -n 34000001,35000000p scripts.txt > 35.txt  sed -n 35000001,36000000p scripts.txt > 36.txt  sed -n 36000001,37000000p scripts.txt > 37.txt  sed -n 37000001,38000000p scripts.txt > 38.txt  sed -n 38000001,39000000p scripts.txt > 39.txt  sed -n 39000001,40000000p scripts.txt > 40.txt  sed -n 40000001,41000000p scripts.txt > 41.txt  sed -n 41000001,42000000p scripts.txt > 42.txt  sed -n 42000001,43000000p scripts.txt > 43.txt  sed -n 43000001,44000000p scripts.txt > 44.txt  sed -n 44000001,45000000p scripts.txt > 45.txt  sed -n 45000001,46000000p scripts.txt > 46.txt  sed -n 46000001,47000000p scripts.txt > 47.txt  sed -n 47000001,48000000p scripts.txt > 48.txt  sed -n 48000001,49000000p scripts.txt > 49.txt  sed -n 49000001,50000000p scripts.txt > 50.txt  sed -n 50000001,51000000p scripts.txt > 51.txt  sed -n 51000001,52000000p scripts.txt > 52.txt  sed -n 52000001,53000000p scripts.txt > 53.txt  sed -n 53000001,54000000p scripts.txt > 54.txt  sed -n 54000001,55000000p scripts.txt > 55.txt  sed -n 55000001,56000000p scripts.txt > 56.txt  sed -n 56000001,57000000p scripts.txt > 57.txt  sed -n 57000001,58000000p scripts.txt > 58.txt |

|  |
| --- |
| #!/bin/bash  #PBS -l nodes=1:ppn=16  #PBS -l walltime=00:12:00:00  #PBS -N gp\_clinical  cd $PBS\_O\_WORKDIR  #Shell script for Blue Crystal  #Create 124 datasets from gp\_clinical.txt file for loading into Stata (1,000,000 rows each)  sed -n 1,1000000p gp\_clinical.txt > 1.txt  sed -n 1000001,2000000p gp\_clinical.txt > 2.txt  sed -n 2000001,3000000p gp\_clinical.txt > 3.txt  sed -n 3000001,4000000p gp\_clinical.txt > 4.txt  sed -n 4000001,5000000p gp\_clinical.txt > 5.txt  sed -n 5000001,6000000p gp\_clinical.txt > 6.txt  sed -n 6000001,7000000p gp\_clinical.txt > 7.txt  sed -n 7000001,8000000p gp\_clinical.txt > 8.txt  sed -n 8000001,9000000p gp\_clinical.txt > 9.txt  sed -n 9000001,10000000p gp\_clinical.txt > 10.txt  sed -n 10000001,11000000p gp\_clinical.txt > 11.txt  sed -n 11000001,12000000p gp\_clinical.txt > 12.txt  sed -n 12000001,13000000p gp\_clinical.txt > 13.txt  sed -n 13000001,14000000p gp\_clinical.txt > 14.txt  sed -n 14000001,15000000p gp\_clinical.txt > 15.txt  sed -n 15000001,16000000p gp\_clinical.txt > 16.txt  sed -n 16000001,17000000p gp\_clinical.txt > 17.txt  sed -n 17000001,18000000p gp\_clinical.txt > 18.txt  sed -n 18000001,19000000p gp\_clinical.txt > 19.txt  sed -n 19000001,20000000p gp\_clinical.txt > 20.txt  sed -n 20000001,21000000p gp\_clinical.txt > 21.txt  sed -n 21000001,22000000p gp\_clinical.txt > 22.txt  sed -n 22000001,23000000p gp\_clinical.txt > 23.txt  sed -n 23000001,24000000p gp\_clinical.txt > 24.txt  sed -n 24000001,25000000p gp\_clinical.txt > 25.txt  sed -n 25000001,26000000p gp\_clinical.txt > 26.txt  sed -n 26000001,27000000p gp\_clinical.txt > 27.txt  sed -n 27000001,28000000p gp\_clinical.txt > 28.txt  sed -n 28000001,29000000p gp\_clinical.txt > 29.txt  sed -n 29000001,30000000p gp\_clinical.txt > 30.txt  sed -n 30000001,31000000p gp\_clinical.txt > 31.txt  sed -n 31000001,32000000p gp\_clinical.txt > 32.txt  sed -n 32000001,33000000p gp\_clinical.txt > 33.txt  sed -n 33000001,34000000p gp\_clinical.txt > 34.txt  sed -n 34000001,35000000p gp\_clinical.txt > 35.txt  sed -n 35000001,36000000p gp\_clinical.txt > 36.txt  sed -n 36000001,37000000p gp\_clinical.txt > 37.txt  sed -n 37000001,38000000p gp\_clinical.txt > 38.txt  sed -n 38000001,39000000p gp\_clinical.txt > 39.txt  sed -n 39000001,40000000p gp\_clinical.txt > 40.txt  sed -n 40000001,41000000p gp\_clinical.txt > 41.txt  sed -n 41000001,42000000p gp\_clinical.txt > 42.txt  sed -n 42000001,43000000p gp\_clinical.txt > 43.txt  sed -n 43000001,44000000p gp\_clinical.txt > 44.txt  sed -n 44000001,45000000p gp\_clinical.txt > 45.txt  sed -n 45000001,46000000p gp\_clinical.txt > 46.txt  sed -n 46000001,47000000p gp\_clinical.txt > 47.txt  sed -n 47000001,48000000p gp\_clinical.txt > 48.txt  sed -n 48000001,49000000p gp\_clinical.txt > 49.txt  sed -n 49000001,50000000p gp\_clinical.txt > 50.txt  sed -n 50000001,51000000p gp\_clinical.txt > 51.txt  sed -n 51000001,52000000p gp\_clinical.txt > 52.txt  sed -n 52000001,53000000p gp\_clinical.txt > 53.txt  sed -n 53000001,54000000p gp\_clinical.txt > 54.txt  sed -n 54000001,55000000p gp\_clinical.txt > 55.txt  sed -n 55000001,56000000p gp\_clinical.txt > 56.txt  sed -n 56000001,57000000p gp\_clinical.txt > 57.txt  sed -n 57000001,58000000p gp\_clinical.txt > 58.txt  sed -n 58000001,59000000p gp\_clinical.txt > 59.txt  sed -n 59000001,60000000p gp\_clinical.txt > 60.txt  sed -n 60000001,61000000p gp\_clinical.txt > 61.txt  sed -n 61000001,62000000p gp\_clinical.txt > 62.txt  sed -n 62000001,63000000p gp\_clinical.txt > 63.txt  sed -n 63000001,64000000p gp\_clinical.txt > 64.txt  sed -n 64000001,65000000p gp\_clinical.txt > 65.txt  sed -n 65000001,66000000p gp\_clinical.txt > 66.txt  sed -n 66000001,67000000p gp\_clinical.txt > 67.txt  sed -n 67000001,68000000p gp\_clinical.txt > 68.txt  sed -n 68000001,69000000p gp\_clinical.txt > 69.txt  sed -n 69000001,70000000p gp\_clinical.txt > 70.txt  sed -n 70000001,71000000p gp\_clinical.txt > 71.txt  sed -n 71000001,72000000p gp\_clinical.txt > 72.txt  sed -n 72000001,73000000p gp\_clinical.txt > 73.txt  sed -n 73000001,74000000p gp\_clinical.txt > 74.txt  sed -n 74000001,75000000p gp\_clinical.txt > 75.txt  sed -n 75000001,76000000p gp\_clinical.txt > 76.txt  sed -n 76000001,77000000p gp\_clinical.txt > 77.txt  sed -n 77000001,78000000p gp\_clinical.txt > 78.txt  sed -n 78000001,79000000p gp\_clinical.txt > 79.txt  sed -n 79000001,80000000p gp\_clinical.txt > 80.txt  sed -n 80000001,81000000p gp\_clinical.txt > 81.txt  sed -n 81000001,82000000p gp\_clinical.txt > 82.txt  sed -n 82000001,83000000p gp\_clinical.txt > 83.txt  sed -n 83000001,84000000p gp\_clinical.txt > 84.txt  sed -n 84000001,85000000p gp\_clinical.txt > 85.txt  sed -n 85000001,86000000p gp\_clinical.txt > 86.txt  sed -n 86000001,87000000p gp\_clinical.txt > 87.txt  sed -n 87000001,88000000p gp\_clinical.txt > 88.txt  sed -n 88000001,89000000p gp\_clinical.txt > 89.txt  sed -n 89000001,90000000p gp\_clinical.txt > 90.txt  sed -n 90000001,91000000p gp\_clinical.txt > 91.txt  sed -n 91000001,92000000p gp\_clinical.txt > 92.txt  sed -n 92000001,93000000p gp\_clinical.txt > 93.txt  sed -n 93000001,94000000p gp\_clinical.txt > 94.txt  sed -n 94000001,95000000p gp\_clinical.txt > 95.txt  sed -n 95000001,96000000p gp\_clinical.txt > 96.txt  sed -n 96000001,97000000p gp\_clinical.txt > 97.txt  sed -n 97000001,98000000p gp\_clinical.txt > 98.txt  sed -n 98000001,99000000p gp\_clinical.txt > 99.txt  sed -n 99000001,100000000p gp\_clinical.txt > 100.txt  sed -n 100000001,101000000p gp\_clinical.txt > 101.txt  sed -n 101000001,102000000p gp\_clinical.txt > 102.txt  sed -n 102000001,103000000p gp\_clinical.txt > 103.txt  sed -n 103000001,104000000p gp\_clinical.txt > 104.txt  sed -n 104000001,105000000p gp\_clinical.txt > 105.txt  sed -n 105000001,106000000p gp\_clinical.txt > 106.txt  sed -n 106000001,107000000p gp\_clinical.txt > 107.txt  sed -n 107000001,108000000p gp\_clinical.txt > 108.txt  sed -n 108000001,109000000p gp\_clinical.txt > 109.txt  sed -n 109000001,110000000p gp\_clinical.txt > 110.txt  sed -n 110000001,111000000p gp\_clinical.txt > 111.txt  sed -n 111000001,112000000p gp\_clinical.txt > 112.txt  sed -n 112000001,113000000p gp\_clinical.txt > 113.txt  sed -n 113000001,114000000p gp\_clinical.txt > 114.txt  sed -n 114000001,115000000p gp\_clinical.txt > 115.txt  sed -n 115000001,116000000p gp\_clinical.txt > 116.txt  sed -n 116000001,117000000p gp\_clinical.txt > 117.txt  sed -n 117000001,118000000p gp\_clinical.txt > 118.txt  sed -n 118000001,119000000p gp\_clinical.txt > 119.txt  sed -n 119000001,120000000p gp\_clinical.txt > 120.txt  sed -n 120000001,121000000p gp\_clinical.txt > 121.txt  sed -n 121000001,122000000p gp\_clinical.txt > 122.txt  sed -n 122000001,123000000p gp\_clinical.txt > 123.txt  sed -n 123000001,124000000p gp\_clinical.txt > 124.txt |

|  |
| --- |
| \*Import the script files and format them  cd [SOMEWHERE]  import delimited "x\1.txt", clear  \*Format event\_dt  rename issue\_date date  gen issue\_date = date(date,"DMY"), a(date)  format issue\_date %td  drop date  save "y\1.dta", replace  forvalues i = 2/58 {  import delimited "x\\`i'.txt", clear  rename v1 eid  rename v2 data\_provider  rename v3 issue\_date  rename v4 read\_2  rename v5 bnf\_code  rename v6 dmd\_code  rename v7 drug\_name  rename v8 quantity    rename issue\_date date  gen issue\_date = date(date,"DMY"), a(date)  format issue\_date %td  drop date    save "y\\`i'.dta", replace  } |

|  |
| --- |
| \*Import the clinical files and format them  cd [SOMEWHERE]  import delimited "x\1.txt", clear  \*Format event\_dt  rename event\_dt date  gen event\_dt = date(date,"DMY"), a(date)  format event\_dt %td  drop date  save "y\1.dta", replace  forvalues i = 2/124 {  import delimited "x\\`i'.txt", clear  rename v1 eid  rename v2 data\_provider  rename v3 event\_dt  rename v4 read\_2  rename v5 read\_3  rename v6 value1  rename v7 value2  rename v8 value3    rename event\_dt date  gen event\_dt = date(date,"DMY"), a(date)  format event\_dt %td  drop date    save "y\\`i'.dta", replace  } |



Alt text: Also, reading 500 coronavirus papers in a row and not sleeping? Probably not great for you either, but I haven't found any studies confirming that yet. I'll keep looking.

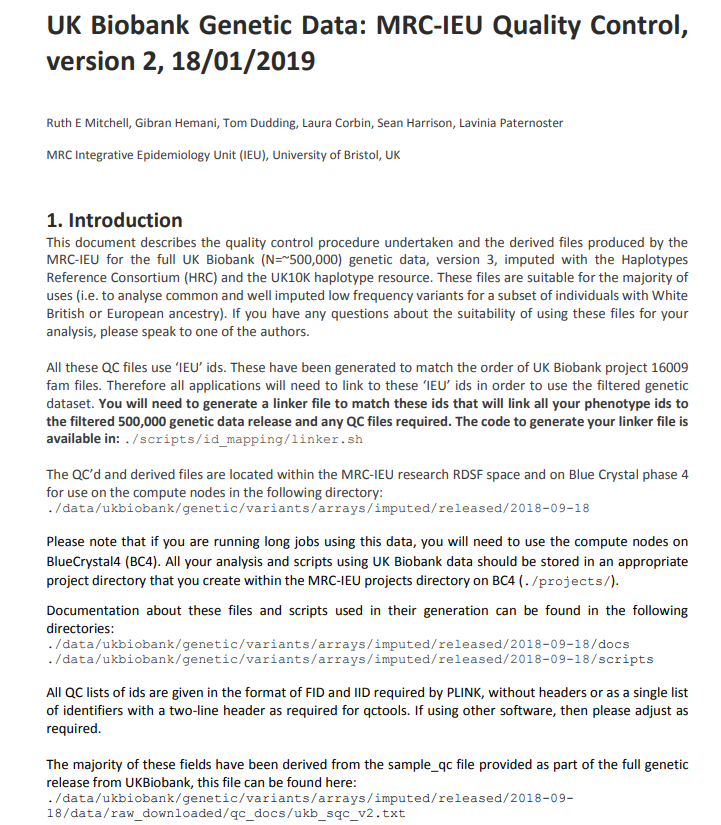
URL: <https://xkcd.com/2281/>

# 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 this code, which you can copy and paste straight into putty:

|  |
| --- |
| ## Script to generate a linker file to the filtered genetic data of UK Biobank ##  # Add in your UK Biobank phenotype application number #  fam\_file="[ENTER YOUR FAM FILE NAME HERE, including the .fam bit]"  #Extract 'IEU' ids from the fam file  awk '{print $1}' /projects/MRC-IEU/research/data/ukbiobank/genetic/variants/arrays/imputed/released/2018-09-18/data/id\_mapping/data.fam > data.ieu.fam  #Extract your phenotype ids - this is available to download alongside your project's phenotypic data  awk '{print $1}' ${fam\_file} > data.app.fam  #load R  module load languages/R-3.4.1-ATLAS  #Start R  R  #Read in the extracted ids  ieu <- read.table('data.ieu.fam', col.names= 'ieu')  app <- read.table('data.app.fam', col.names = 'app')  #column bind them together as the order is the same to create the linker file  linker <- cbind(ieu, app)  #save the linker file  write.table(linker, file = 'linker.csv', row.names=F, col.names=T, quote=F, sep=",")  q(save = "no")  module rm languages/R-3.4.1-ATLAS |

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  inbgen=${bgen\_pattern/CHROM/$chrom}  inbgenidx=${bgen\_index\_pattern/CHROM/$chrom}  **bgenix -g $inbgen -i $inbgenidx -incl-rsids $snp\_list > $temp\_geno\_prefix.$chrom.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\*  module rm apps/bgen-1.01 |

* **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  module rm apps/plink-2.00 |

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 --clump-kb 10000 --out clumped\_snps  tr -s [:blank:] < clumped\_snps.clumped | cut -f 4 -d " " > clumped\_snps.txt  module rm apps/plink-1.90 |

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:

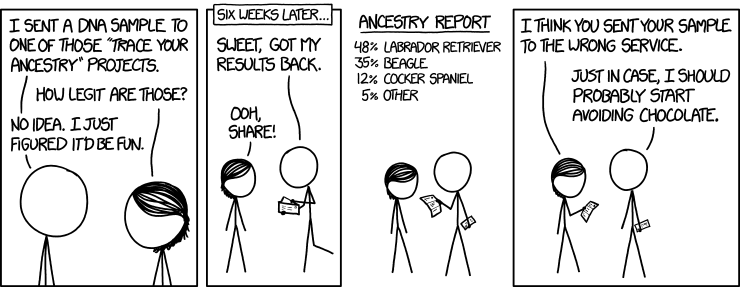
1. Loads the Bgenix and Plink 2.0 modules
2. Iterates through the 23 chromosomes
3. Uses Bgenix to extract the relevant SNPs
4. Combine the SNPs into a single BGEN file & removes temporary bgen files
5. Use plink’s --score option with a text file containing a list of SNPs, their effect alleles, and the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

|  |
| --- |
| module load apps/bgen-1.0.1  module add apps/plink-2.00  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  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  inbgen=${bgen\_pattern/CHROM/$chrom}  inbgenidx=${bgen\_index\_pattern/CHROM/$chrom}  bgenix -g $inbgen -i $inbgenidx -incl-rsids $snp\_list > $temp\_geno\_prefix.$chrom.**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\*  plink --bgen instruments.bgen --score $snp\_list\_alleles --out score  module rm apps/bgen-1.0.1  module rm apps/plink-2.00 |

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 snp\_list.txt file must have a list of rsIDs, no quote and no header, as above.

The score from plink is an average across all alleles, so needs to be multiplied by the number of non-missing SNPs (also given in the .sscore file) to get the sum of betas\*dosage. There are, apparently, options to produce a sum, rather than an average, but I can’t find out how to do this with plink 2.0.



Alt text: Plus, now I know that I have risk factors for elbow dysplasia, heartworm, parvo, and mange.

URL: <https://xkcd.com/1706/>

Creating a Genetic Risk Score Using the mrbase\_grs R Function

The mrbase\_grs function creates genetic risk scores (GRS) using UK Biobank data given a list of studies or traits in MR Base. It creates 4 files that need to be copied to the Blue Crystal RDSF, and additional files are required if using proxies or clumping is required. The SNPs in the GRS are weighted by the beta coefficient reported by the GWAS found in MR Base.

## 1. Options

|  |  |
| --- | --- |
| Argument | Description |
| **output=NULL,** | The output of the function, only mandatory argument. One of:   * subcategories * traits * studies * snps * code |
| category=NULL  subcategory=NULL  trait=NULL  population=NULL  sex=NULL  mr=NULL  samplesize=0  notstudies=NULL  studies=NULL  p=5e-08 | These options limit the retrieved studies. Lists as vectors can be specified.  *Category*, *subcategory* and *trait* limit study focus, and *population*, *sex* (male, female, male and female), *mr* (MR Base – does the study have data for all SNPs that allow MR?) and *sample size* (>= specified number) limit the study population.  *Notstudies* and *studies* limit by individual studies.  *P* limits SNPs to those with a P value less than specified (when downloading from MR Base, not in specified data frames) |
| exposure\_dat=NULL  exposure\_file = NULL | *exposure\_dat* is a data frame containing all SNP data required to create a GRS, which needs to be in the correct format for MR Base. The *exposure\_dat* file is created using MR Base if not specified, so this option is only relevant if you have a list of SNPs and effects etc.  *exposure\_file* is a csv containing all the *exposure\_dat* data, meaning you don’t need to read the data into R for the function to create the relevant code. Note the headings need to be in the correct format for MR Base, the same as *exposure\_dat* |
| proxies=TRUE  r2=0.8 | If *proxies* are specified, the program will search for proxies for SNPs not in UK Biobank. The *r2* gives a lower limit for the *r2*value for proxies to be allowed (SNPs not in UK Biobank without a proxy above this threshold will not be used in the GRS). The lower *r2* limit is 0.6 |
| clump=TRUE  clumped=TRUE  clump\_kb=10000  clump\_r2=0.001 | *clump* alters behaviour of the *extract­\_instruments* function in MR Base. If *TRUE*, MR Base will download pre-clumped SNPs.  *clumped* tells the program that data specified with *exposure\_dat* or *exposure\_file* has been clumped. Set to false if the data needs clumping.  The code will automatically set *clumped* to false if it is forced to download unclumped data from MR Base, which happens when too many SNPs are below the P value threshold – you don’t need to do anything if this happens, but it will slow down the code  The clump\_kb and clump\_r2 are options for the clumping in Plink/MR Base, set to the MR Base default |
| gwas="biggest", | An option for determining which studies to use when gathering SNPs. Options are “**biggest**” or “**all**”. *Biggest* extracts SNPs from the study with the most participants. *All* uses all studies for a particular trait, removing duplicate SNPs by selecting the SNP from the largest study. Note: when more than one study is used, the data will be re-clumped |
| suffix=”” | Adds a suffix to any files created, allowing for a) more informative file names, e.g. suffix = “\_05” and “\_08” to distinguish between P value thresholds. Allows running of several scripts simultaneously in Blue Crystal |
| snpstats\_file="…"  ld\_file="…"  eur\_file = “eur” | File names of required files. The SNP stats file will be created if not present, but the LD and EUR files must be copied from the project folder, or otherwise created/downloaded. The LD file is used for finding proxies, and the EUR files are used in clumping. Both come from 1000 genomes (EUR is Europeans only). |
| keep\_files=FALSE | If *TRUE*, will not delete selected intermediate files. If you want ALL intermediate files, you can always edit the code generated by the program. If *FALSE*, only necessary files will be kept |
| ipd=FALSE | If *TRUE*, generates a csv containing dosages of all SNPs requested for all participants in UK Biobank. Allows for MR analyses in addition to GRS analyses. IPD = individual participant data |
| maf = 0.42 | Used when harmonising MR Base with 1000 genomes and UK Biobank. This is the lower limit for minor allele frequency can be to determine intermediate frequency palindromic SNPs. Intermediate frequency palindromic SNPs cannot be reliably harmonised, EXCEPT when the strand can be determined for all other SNPs from the same study, in which case strand can be inferred and the MAF becomes irrelevant |
| plink\_grs=FALSE | If *TRUE*, the program will use plink to generate the GRS, not R. This gets around memory issues if using a large (>10,000) number of SNPs. This option ONLY WORKS for single traits – do not use this option when specifying more than one trait, you’ll still only get one (incomprehensible) GRS out. If you have specified many traits/studies and run into memory issues, split the traits/studies between more than one mrbase\_grs command and use prefixes to distinguish between them. |
| bc=4 | The version of Blue Crystal you want the code to run on. Default is 4 (Blue Crystal 4), where you have to submit the job to the queue. The alternative is 3 (Blue Crystal 3), where you have to run the job on the login node. Blue Crystal 4 is essential if you’re using many SNPs, but it’s also unbelievably fast for small PRS. |
| bgen\_folder="" | The folder to take the UK Biobank data from. This code will likely work fine if you specify a different folder containing the SAME type of data - .bgen and .bgen.bgi for 23 chromosomes. Note – you need access to this folder to use this program. |

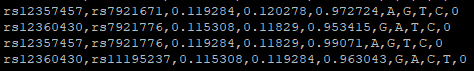
## 2. Additional Files

To run the code fully, additional files are required in your home directory. The first two files can be copied from my project folder (**M:\projects\ieu2\p6\006\working\data**, email [sean.harrison@bristol.ac.uk](mailto:sean.harrison@bristol.ac.uk) for access). The final file (the SNPStats.txt file) can be created by the code, or copied with the other two files.

### 1. 1000 genomes LD file – searching for proxies

This file shows the SNPs in linkage disequilibrium (R2 above 0.6) for all SNPs in 1000 genomes. It’s used to find proxies for SNPs that are in the original GWAS, but aren’t in the target dataset. It needs to show the original SNP, proxy SNP, EAF of both SNPs, R2 between the SNPs, and the alleles of both SNPs.

Called **1000\_genomes\_ld.csv.gz** by default.



### 2. European 1000 genomes data (bed, bim, fam) – clumping SNPs

These 3 files (bed, bim and fam) are raw genetic data for the European subsample of 1000 genomes. They are used to clump SNPs, removing any SNPs with R2 values of more than a threshold (0.001).

Called **eur.bed, eur.bim** and **eur.fam** by default.

### 3. SNPstats file

This file is a record of all SNPs in UK Biobank, and includes rsID, major/minor alleles, major allele frequency and info score, among other things. This file is used to see whether a SNP is in UK Biobank, and if so, what the major/minor alleles are for harmonisation.

Called **SNPstats.txt** by default. If not present, it will be created from the individual chromosome SNPstats files.

## 3. Code Procedures

### Step 0. Creating a list of relevant studies in MR Base given options specified

Creates a list of studies, part of most outputs.

1. Grab the available outcomes from MR BASE:
   1. **mrbase\_grs\_ao = available\_outcomes()**
2. Replace some missing values in sample size and consortium
3. Restrict the available outcomes by specified options (e.g. sex = “male”)

### Output 1. subcategories

Returns a list of subcategories, and the number of traits and studies in each subcategory.

1. Step 0
2. Create a list of subcategories from the outcomes
3. Return a dataframe showing the number of studies and traits for each subcategory

### Output 2. traits

Returns a list of traits, the number of studies in each trait, and the subcategory to which the trait belongs.

1. Step 0
2. Create a list of traits from the outcomes
3. Return a dataframe showing the number of studies for each trait, and which subcategory the trait is within

### Output 3. studies

Returns a list of studies. The information is a slimmer version of the output of *available\_outcomes* in MR BASE.

1. Step 0.
2. Return the outcomes dataframe, removing selected columns

### Output 4. snps

Returns a list of SNPs from relevant studies. This includes a merger of the info in *available\_outcomes* and *extract\_instruments* in MR Base.

1. If studies are NOT specified:
   1. Step 0
   2. Create a list of studies to be included
2. Else:
   1. Check if the studies listed are numeric, if so that is the list of studies
3. If the list of studies is not empty, extract the instruments using the P and clumping options specified:
   1. **mrbase\_grs\_instruments = extract\_instruments(outcomes=mrbase\_grs\_studies, p1 = p)**
4. Return these instruments, merged with study level information

### Output 5. code

This generates and saves four files necessary to create a GRS for each unique trait

1. If *exposure\_dat* or *exposure\_file* are specified, nothing is downloaded from MR Base
2. Else if *traits* are specified:
   1. Step 0
   2. Create a list of studies to be included
   3. If GWAS option “biggest” is specified, use the biggest study only, otherwise include all studies for each trait
   4. Create **exposure\_dat.csv** using the resulting SNPs
3. Else if *studies* are specified:
   1. Check the list of studies is numeric
   2. Step 0
   3. Limit the available outcomes to specified studies
   4. Create **exposure\_dat.csv** using the resulting SNPs
4. The **script.R** script is created, which is the main part of this process and will be described below
5. The **run.sh** script is created, which adds relevant modules to Blue Crystal to allow script.R to run
6. The **harmonise.R** script is created, which is used to help the harmonisation step(s)
7. Note: all files will have suffixes (if specified), except **harmonise.R** which can be used for all programs, unless you change the value of the MAF.

## 4. Running the Program

To use the mrbase\_grs function, you first need to copy it from my project folder (M:\projects\ieu2\\_working\IEU2\_P6\_006\data\R) or Github (<https://github.com/sean-harrison-bristol/UK_Biobank_PRS>) to a folder of your choice.

Once you have the code, import it into R:

**source(“M:/projects/ieu2/\_working/IEU2\_P6\_006/data/Code/mrbase\_grs\_v3.01.R”)**

The syntax for the command is above, but some example code is given at the bottom of this document. The first time you use the code, you’ll need to login to google – this is a check the MR Base uses, not mrbase\_grs, and logging in will create an authorisation file in the folder you called the program from. Any packages used by mrbase\_grs will be installed and added if not already in your version of R.

Once you have the four files created by mrbase\_grs, follow these steps to complete the process. You must have logins for WinSCP and Putty, and access to the UK Biobank genetic data on either Blue Crystal 3 or 4.

1. Copy the files in “Additional files” in my project data folder to your home directory on Blue Crystal, as well as the four files created using the mrsbase\_grs function. You can use WinSCP for this.
2. Type these commands into Putty (when in your home directory, type **pwd** to find out where you are, and **cd** to change directory):
   1. **chmod +x ru\* sc\***
   2. **dos2unix ru\* sc\***
3. This will give the **run.sh** and **script.R** files executable permission, and convert them from DOS (Windows) to Unix (Linux). These steps are mandatory if working on a Windows computer, since carriage returns are differently specified between the systems.
4. If using Blue Crystal 4 (the default):
   1. Type **sbatch run.sh** (or whatever the run script is called if you used a suffix). This will send the run script to the Blue Crystal 4 queuing system. I have entered reasonable values for the nodes, tasks per node, CPUs per task, time and memory, but if you are finding the code needs more time, increase the value of **--time** in the run script, and if you run out of memory, increase the value of **--mem**.
5. If using Blue Crystal 3:
   1. Use the login node, and type: **./run.sh** to create the PRS – using **qsub** doesn’t work as it can’t access the BGEN files
   2. **NOTE**: if you have a large PRS with many SNPs, you are much better off using Blue Crystal 4, partly because it’ll be quicker, partly because you may get an angry email from IT saying “stop using the login node!”
6. The scripts will produce a file called **grs.csv**, which contains a list of IDs and a list of GRS for all traits you specified. A file called **exposure\_dat\_harmonised.csv** is also created, and this contains details for all SNPs used in the GRS, including whether or not a proxy was used, the effect allele, betas etc. By default, an individual participant data file will also be produced containing dosage data for all SNPs specified, **snp\_ipd.raw**.
7. All participants in UK Biobank with genetic data are included – remember to exclude individuals if you only want, for example, only White British individuals, or the set of minimally related individuals. See here for more information:
   1. <https://data.bris.ac.uk/datasets/3074krb6t2frj29yh2b03x3wxj/UK%20Biobank%20Genetic%20Data_MRC%20IEU%20Quality%20Control%20version%201.pdf>

## 5. Additional Notes

### Proxies

The program checks which SNPs in *exposure\_dat* are in UK Biobank using the SNP stats file. For any SNPs not in UK Biobank, proxies are searched for using a file from 1000 genomes. The R2 value for these proxies are checked against the minimum (0.8 by default), and then these are searched for in UK Biobank. Proxy SNPs remaining that are in UK Biobank are ranked by their R2, and the top hit is selected as the proxy.

Proxies are marked in the **exposure\_dat\_harmonised.csv** file, with the proxy SNP rs ID in the “SNP” column, and original SNP rs ID in the “Original SNP” column. The r2 value is also included. To not search for proxies, use the proxies=FALSE option.

### Harmonisation

MR Base SNPs must be harmonised with SNPs in 1000 genomes and UK Biobank. There are three main issues in harmonisation, all of which are dealt with in the program:

1. The effect allele could be the other allele in different studies, e.g. in one study, the SNP A/C could have A as the effect allele, whereas in another the C could have been the effect allele
2. The strand could be different between studies, i.e. the forward strand could be used in one study, and the reverse in another. For example, on the forward strand the SNP is A/C, but on the reverse strand, the SNP is T/G
3. Palindromic SNPs make it impossible to tell whether SNPs are on the forward or negative strand, and thus which is the effect allele. Palindromic SNPs are either A/T or C/G, and thus appear the same whether on the forward or reverse strands. However, when the minor allele frequency is low (e.g. <0.3) you can think of the SNPs as [A/C/T/G]minor and [A/C/T/G]Major, so the forward and reverse strands are now identifiable. So long as the frequencies are relatively similar between studies, you can work out the effect allele.

The program gets around the final problem by seeing whether all the non-palindromic SNPs in one study are on the same strand as the other study (or whether they are all on the other strand). If so, then the minor allele frequency is not required to determine whether a palindromic SNP is on a different strand. If not, then any palindromic SNPs with an intermediate minor allele frequency (>0.3 by default) will be discarded.

### Clumping

Clumping is done using plink 1.90 and the European participants from 1000 genomes:

**plink -bfile eur --clump unclumped\_snps.txt --clump-p1 1 --clump-p2 1 --clump-r2 0.001 --clumb-kb 10000 --out clumped\_snps**

**tr -s [:blank:] < clumped\_snps.clumped | cut -f 4 -d " " > clumped\_snps.txt**

### Bgenix and Plink

Bgenix is used to select SNPs from the UK Biobank .bgen files, before passing them to Plink 2.00 to create dosages:

**bgenix -g $inbgen -i $inbgenidx -incl-rsids $snp\_list > $temp\_geno\_prefix.$chrom\_padd.bgen**

**plink2 –bgen instruments.bgen --hard-call-threshold 0.4999 --export A --out instruments**

### Triallelic SNPs

About 50,000 SNPs are triallelic, so have two different other alleles or effect alleles. These mess with the code too much to be included, so are excluded immediately. Also note that Plink doesn’t work with triallelic SNPs.

### Field names

If you are specifying an *exposure\_dat* or *exposure\_file*, then ensure the headings are identical to those required by MR Base. Specifically, you require:

* SNP (string)
* effect\_allele.exposure (string)
* other\_allele.exposure (string)
* eaf.exposure (numeric)
* beta.exposure (numeric)
* pval.exposure (numeric, optional unless clumping is necessary)
* trait (string, optional but desirable)

### Errors

Memory errors can occur if the number of SNPs is too great. I think this is an issue with R on Blue Crystal, i.e. there’s too many SNPs in memory at once. In any case, if using a single trait, use *plink\_grs=TRUE*, otherwise split the traits into manageable chunks.

Other errors may result from using this program in a way I haven’t anticipated. I’ve put in some controls to stop obvious errors in R, but can’t account for everything. You can always email me if you are having trouble, but if you’re doing something advanced, feel free to edit any code to suit your purposes, just let me know whether it’s something I should implement.

## 6. Examples

Suppose that initially, you don’t know which traits are available in MR Base, but knew you only wanted to look at Europeans:

**traits = mrbase\_grs(output="traits", population = "European")**

**write.csv(traits,file="traits.csv")**

This creates a CSV file containing all the traits in MR Base, and you can select traits from there to include. For this example, let’s say you chose “Body mass index” and “Height”, and now want to have a look at which studies in MR Base look at these traits (but don’t include UK Biobank in the GWAS):

**traits = c("Body mass index", "Height")**

**studies = mrbase\_grs(output="studies", trait = traits, population = "European")**

**studies = studies[studies$consortium != "UK Biobank",]**

**write.csv(studies,file="Studies.csv")**

You can now see there are 13 studies looking at height and BMI. However, most of them are from the same consortia (GIANT and ECC). You should always check those consortia online to see whether UK Biobank was included, but in these studies, it wasn’t. You can now choose which studies to include (from the ID field in the studies data frame): in this case, you go with studies 89 and 835.

**dat8 = mrbase\_grs(output="code",studies = c(89,835))**

By default, this will give us the dosage data for all SNPs as well as the GRS, so you can run MR sensitivity analyses along with the GRS analysis.

This code generates the four standard files, which you copy across to Blue Crystal using WinSCP. You need to add executable permission to the script and run files, convert the code from DOS to UNIX, then submit the run script using **sbatch**. This produces the grs.csv and snp\_ipd.raw files you can then use in your analysis. All participants have GRS, so you may want to restrict them before any analysis, e.g. to white British participants.

If you want all SNPs with a P value below 5e-05, you can type:

**dat5 = mrbase\_grs(output="code", studies = c(89,835), p=5e-05,suffix = "\_05")**

If you then change your mind and don’t want to create the IPD file, you can change the code without redownloading the SNPs:

**dat5x = mrbase\_grs(output="code",exposure\_dat = dat5,ipd=FALSE,suffix = "\_05x")**

If you want to create a P=1 GRS, using ALL the SNPs measured in a GWAS, you can download the relevant genetic data from the GIANT website, rename any fields that don’t match, and specify that dataset as the exposure\_dat.

This will work for any GWAS - you just need to have sufficient data in the correct format.

## 7. Statement for Papers

Information for SNPs associated with [trait[s]] in previous GWAS were downloaded from MR-Base [ref] using the R statistical package, including rs IDs, effect estimates, effect alleles and relevant study information. [If **proxies** used] The 1000 genomes project was used to find proxy SNPs in LD (R2 above 0.8) with SNPs not found in UK Biobank. The SNPs from MR-Base were harmonised with the SNPs from UK Biobank, aligning the effect estimates and alleles. [If **clumping** used] SNPs were clumped used the European subsample of the 1000 genomes project, with R2 < 0.001 and a window of 10,000 kb. The genetic risk score[s] were created by multiplying the number of effect alleles for each participant in UK Biobank by the effect estimate of the SNP from MR-Base, then summing across all SNPs associated with each trait.

**-- Feel free to edit the above --**