Manual for `grm_tool.f90`

A fortran program to summarise and filter GRMs

Jisca Huisman

2024-01-10

Description

This program can calculate various summary statistics from (potentially huge) GRMs, and/or filter out pairs with R values above or below a specified threshold. Additional functionality may be added in the future.

Preparations

Requirements

- a Fortran compiler, e.g. gfortran
- gzip (or pigz), to decompress the .grm.gz file

Compilation

Run `gfortran -O3 grm_tool.f90 -o grm_tool` (where you may substitute gfortran by another compiler of your choice)

Input files

- `--in` : input file pair with GRM, as returned by `plink --make-grm-gz` or `gcta --make-grm`. Extensions .grm.id and .grm.gz are added.
- `--only` : individual subset, only pairs where either individual is listed are considered. Text file with a single column with IDs, no header row.
- `-only-among` : as `--only`, but only pairs where *both* individuals are listed are considered. grm

---

Program options

Summary statistics

Currently, the only command line options are to turn calculation off (`--no-summary`) and to specify the outfile (`--summary-out`).

Turning the calculation off potentially allows filtering of GRMs with more individuals, see section 'Method'.

The code to calculate the summary statistics looks (similar to) as follows:

```
  write(42,*)  'min_R      ',  MINVAL(GRM)
  write(42,*)  'max_R      ',  MAXVAL(GRM)
  write(42,*)  'mean_diag  ',  SUM(GRM, MASK=IsDiagonal)/COUNT(IsDiagonal)
  write(42,*)  'sd_diag    ',  SD(GRM, MASK=IsDiagonal)    ! function defined in module Fun
```

```
write(42,*)  'mean_betw ',  SUM(GRM, MASK=(.not. IsDiagonal))/COUNT(.not. IsDiagonal)
write(42,*)  'sd_betw   ',  SD(GRM, MASK=(.not. IsDiagonal))
write(42,*)  'count_1.5_diag ',  COUNT(GRM > 1.5 .and. IsDiagonal)
write(42,*)  'count_5_diag   ',  COUNT(GRM > 5 .and. IsDiagonal)
write(42,*)  'count_0.75_betw ', COUNT(GRM > 0.75 .and. .not. IsDiagonal)
write(42,*)  'count_1.5_betw ',  COUNT(GRM > 1.5 .and. .not. IsDiagonal)
```

and additional commands can be added here relatively safely. The syntax is as follows:

- `write(42,*)` : command to write to unit=42; this 'unit' is opened with a specified file name before all write statements, and is closed after the last write statement.
- a text label, between quotes, written verbatim to the output file
- a comma, separating the text label from the next item to write on that line. Multiple items may be written to each line, each separated by a comma; they will be tab-delimited in the output
- a Fortran command to calculate the specific summary statistic. 'GRM' is the vector with R values, and 'IsDiagonal' an equally long vector denoting if this value is on the diagonal of the matrix or not.

Filtering

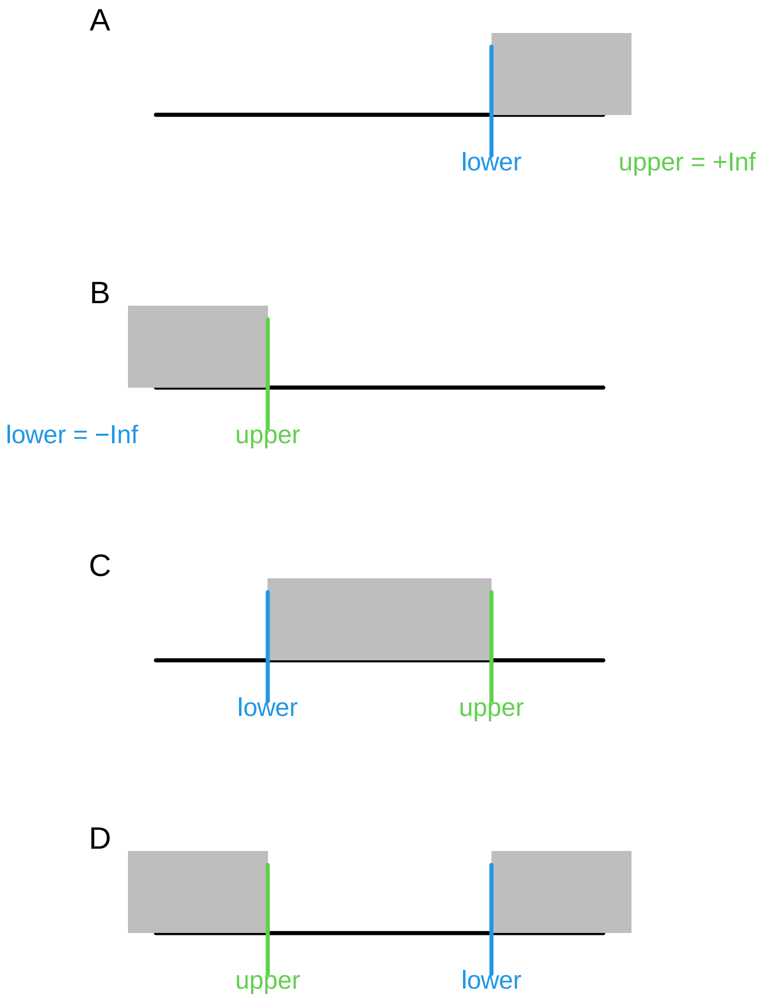
A subset of the GRM can be written to a text file based on their R values, e.g. to check any weird values (e.g. $R > 1.5$ or $R < -1.0$) or to find off-diagonal pairs which may be 1st degree relatives but not duplicates or inbred (e.g. $0.35 < R < 0.65$).

NOTE: This is meant as a tool for data quality control, and I strongly advise against using this as primary method to identify relatives. For that, please use sequoia (https://github.com/JiscaH/sequoia_notR), `find_pairs.f90` (in https://github.com/JiscaH/sequoiaExtra), or other likelihood-based methods.

Separately for the diagonal and off-diagonal ('between') elements, one or two thresholds can be set:

- `diag-lower`, `betw-lower` : the lower threshold, pairs with values above this thresholds are written to the output file;
- `diag-upper`, `betw-upper` : the upper threshold.

One threshold may be left unspecified, and the `upper` threshold may be higher or lower than the `lower` threshold, giving four different possibilities as illustrated as A–D in the figure (grey = selected pairs written to the output).



NOTE: Filtering will currently ignore any values of `+Inf` or `-Inf`, as the thresholds do not default to infinity but to 'HUGE(0D0)', which is the largest 'double precision' value that Fortran can store. Whether those are present in the data can be checked in the summary statistics as

```
write(42,*)  'count_inf ',  COUNT(GRM >= HUGE(0D0))
```

If this is an issue, please let me know as it would be fairly straightforward to change.

Histogram

Counts per histogram bin are calculated with `--hist`, with separate counts for the diagonal and off-diagonal ('between') elements written to a 3-column text file (default hist_counts.txt, specify with `--hist-out`). The first column in the output is the lower bound of each bin. These are equidistant and default from -1.5 to +2.0 with a stepsize of 0.05. These can be adjusted as optional arguments to `--hist`, in which case all 3 must be provided, in the order first, last, step.

The output is (should be) identical to the output of the R command

```
  table(cut(GRM, breaks=seq(first, last, step)))
```

All bins are closed on the right and open on the left, any values <= first or > last are discarded, with a warning. It might therefore be advisable to edit the summary statistics to count the number of values on and off the diagonal exceeding 'last'. Note that the warning will be given only after all the data has been processed, which may take more than an hour for very large GRMs.

The output can e.g. be visualised in R as follows:

```r
H <- read.table('hist_counts.txt', header=TRUE)
# remove superflous head & tail
min_R <- with(H, min(lower_bound[diagonal>0 | between>0]))
max_R <- with(H, max(lower_bound[diagonal>0 | between>0]))
H <- H[H$lower_bound >= min_R & H$lower_bound <= max_R, ]

# plot
plot(H$lower_bound, H$diagonal, type='h', xlim=c(-.4, 1.5),
    main='Diagonal', ylab='Count', xlab='R' )
# - or -
barplot(H$between, space=0, names.arg=H$lower_bound,
        main='Off-diagonal', ylab='Count', xlab='R')
# - or -
HH <- list(breaks = c(H$lower_bound,
                      H$lower_bound[nrow(H)] +
                        (H$lower_bound[2]-H$lower_bound[1])),
           counts = H$diagonal,
           xname = 'R diagonal')
class(HH) <- 'histogram'
plot(HH, xlim=c(0, 1.3))
```

Method

The compressed .grm.gz file is decompressed and processed as a continuous stream of data via a so-called 'named pipe', as described at https://genomeek.wordpress.com/2012/05/07/tips-reading-compressed-file-with-fortran-and-named-pipe/

The speed of the program is limited by the decompression speed. Whereas compression of data can be done in parallel on multiple threads, decompression cannot (according to https://zlib.net/pigz/pigz.pdf)

For the filtering, the data is taken from the stream, checked against the specified thresholds, written to file (or not), and discarded. The amount of computer memory required by the program does thus not increase with the size of the GRM, although the output file may get very large depending on the thresholds used. For the summary statistics and `--hist` all R values are (currently) stored, which may be a very large vector.

Example data

A small mock GRM is available to try out this program and its various setting. The files 'griffin.grm.gz' and 'griffin.grm.id' contain a GRM generated from SNP data simulated from 'Ped_griffin' in the sequoia R package. It includes 200 individuals, including 2 inbred pairs (R>0.75).
