Non-Parametric Shrinkage (NPS)

NPS is a non-parametric polygenic risk prediction algorithm described in Chun et al. (2018) (preprint). NPS starts with a set of summary statistics in the form of SNP effect sizes from a large GWAS cohort. It then removes the correlation structure across summary statistics arising due to linkage disequilibrium and applies a piecewise linear interpolation on conditional mean effects. The conditional mean effects are estimated by partitioning-based non-parametric shrinkage algorithm using a training cohort with individual-level genotype data.

For citation:

Chun et al. Non-parametric polygenic risk prediction using partitioned GWAS summary statistics. BioRxiv 370064, doi: https://doi.org/10.1101/370064 (preprint).
For inquiries on software, please contact:

- Sung Chun (SungGook.Chun@childrens.harvard.edu)
- Nathan Stitziel (nstitziel@wustl.edu)
- Shamil Sunyaev (ssunyaev@rics.bwh.harvard.edu).

The current version is 1.1.0. The followings were changed with this version:

- Command line interface was simplified for the useability improvement.
- Tracy-Widom statistic is used to truncate noisy low-rank projections instead of imposing a preset threshold on eigenvalues.
- GWAS peak selection algorithm was improved to better account for conditionally independent signals.

How to Install

1. Download and unpack NPS package as below (version 1.1.0) (Release Note). Some of NPS codes are optimized in C++ and need to be compiled with GNU C++ compiler (GCC-4.4 or later). This will create two executable binaries, **stdgt** and **grs**, in the top-level NPS directory. **stdgt** is used to convert allelic dosages to standardized genotypes with the mean of 0 and variance of 1. **grs** calculates genetic risk scores using per-SNP genetic effects computed by NPS.

```
2. tar -zxvf nps-1.1.0.tar.gz
3. cd nps-1.1.0/
   make
```

4. The core NPS module was implemented in R and requires R-3.3 or later (available for download at https://www.r-project.org/). Although NPS can run on a standard version of R, we strongly recommend using R linked with a linear algebra acceleration library, such as OpenBLAS, Intel Math Kernel Library (MKL) or Microsoft R open. These libraries can substantially speed up NPS operations.

5. (*Optional*) NPS relies on R modules, pROC and DescTools, to calculate the AUC and Nagelkerke's R^2 statistics, respectively. These modules are optional; if they are not installed, AUC and Nagelkerke's R^2 will not be reported. To enable this feature, please install these packages by running the following on command line:

```
6. Rscript -e 'install.packages("pROC", repos="http://cran.r-project.org")'
```

```
Rscript -e 'install.packages("DescTools", repos="http://cran.r-project.org")'
```

To install the R extensions in the home directory (e.g. ~/R) rather than in the default system path, please run the following instead:

```
Rscript -e 'install.packages("pROC", "~/R", repos="http://cran.r-project.org")'
Rscript -e 'install.packages("DescTools", "~/R", repos="http://cran.r-project.org")'

# Add "~/R" to the local R library path in your login shell's start-up file.
# For example, in case of bash, add the following to .bash_profile or .bashrc:
export R_LIBS="~/R:$R_LIBS"
```

7. Although we provide a command line tool to run NPS on desktop computers without parallelization, we strongly recommend running it on computer clusters, processing all chromosomes in parallel. To make this easier, we provide job script examples for SGE and LSF clusters (See sge/ and lsf/ directories). You may still need to modify the provided job scripts to configure and load necessary modules similarly as in the following example:

```
###
# ADD CODES TO LOAD MODULES HERE
# ---------------------- EXAMPLE ----------------------------
# On clusters running environment modules and providing R-mkl
module add gcc/5.3.0
module add R-mkl/3.3.2

# On clusters running DotKit instead and supporting OpenblasR
use GCC-5.3.0
use OpenblasR
# -----------------------------------------------------------
...
```

The details will depend on specific system configurations.

19. We provide job scripts to help prepare training and validation cohorts for NPS. These scripts require bgen and qctool v2.

Input files for NPS

To run NPS, you need the following set of input files:

1. **GWAS summary statistics.** NPS supports two summary statistics formats: *MINIMAL* and *PREFORMATTED*. Internally, NPS uses PREFORMATTED summary statistics. With real data, however, we generally recommend preparing summary statistics in the MINIMAL format and harmonize them with training genotype data using provided NPS scripts. This will automatically convert the summary statistics file into the PREFORMATTED format (step-by-step instruction).
    - The MINIMAL format is a *tab-delimited* text file with the following seven or eight columns:
        - **chr**: chromosome number. NPS expects only chromosomes 1-22. *Only chromosome numbers are expected.*
        - **pos**: base position of SNP.

- **a1** and **a2**: alleles at each SNP in any order.
- **effal**: effect allele. It should match either a1 or a2 allele.
- **pval**: p-value of association.
- **effbeta**: estimated *per-allele* effect size of *the effect allele*. For case/control GWAS, log(OR) should be used. *DO NOT pre-convert them to effect sizes relative to standardized genotypes. NPS will handle this automatically.*
- **effaf**: (*Optional*) allele frequency of *the effect allele* in the discovery GWAS cohort. If this column is missing, NPS will use the allele frequencies of training cohort instead. Although this is optional, we **strongly recommend including effaf when it is available from a GWAS study.** When this column is provided, NPS will run a QC check for the consistency of the effect allele frequencies between GWAS and training cohort data.

```
chr pos      a1       a2       effal    pval     effbeta  effaf
1   569406   G        A        G        0.8494   0.05191  0.99858
1   751756   C        T        C        0.6996   0.00546  0.14418
1   753405   C        A        C        0.8189   0.00316  0.17332
1   753541   A        G        A        0.8945   0.00184  0.16054
1   754182   A        G        A        0.7920   0.00361  0.18067
1   754192   A        G        A        0.7853   0.00373  0.1809
1   754334   T        C        T        0.7179   0.00500  0.18554
1   755890   A        T        A        0.7516   0.00441  0.17327
1   756604   A        G        A        0.9064   0.00162  0.18202
...
```

- The *PREFORMATTED* format is the native format for NPS. We provide summary statistics of our test cases in this format so that NPS can run on them directly without conversion. This is a *tab-delimited* text file format, and rows must be sorted by chromosome numbers and positions of SNPs. The following seven columns are required:

  - **chr**: chromosome name starting with "chr." NPS expects only chromosomes 1-22. *Chromosomes should be designated by "chr1", ..., "chr22".*
  - **pos**: base position of SNP.
  - **ref** and **alt**: reference and alternative alleles of SNP, respectively.
  - **reffreq**: allele frequency of reference allele in the discovery GWAS cohort.
  - **pval**: p-value of association.
  - **effalt**: estimated *per-allele* effect size of *the alternative allele*. For case/control GWAS, log(OR) should be used. NPS will convert **effalt** to effect sizes relative to *the standardized genotype* internally using **reffreq**.

```
chr pos      ref      alt      reffreq  pval     effalt
chr1         676118   G        A        0.91584  0.7908   0.0012
chr1         734349   G        A        0.90222  0.6989   0.001636
chr1         770886   G        A        0.91708  0.721    0.001627
chr1         785050   G        A        0.1139   0.3353   -0.00381
chr1         798400   G        A        0.8032   0.03301  0.006736
chr1         804759   G        A        0.8837   0.7324   -0.00134
chr1         831489   G        A        0.2797   0.1287   0.004252
chr1         832318   G        A        0.2797   0.4102   0.002304
chr1         836924   G        A        0.7958   0.6591   -0.001374
...
```

  When summary statistics are provided in this format, NPS will not run its automatic data harmonization procedures. The user needs to ensure that the summary statistics file does not include InDels, tri-allelic SNPs or duplicated markers and also

that no SNP in training genotypes files has missing summary statistics. If these requirements are violated, NPS will report an error and terminate.

2. **Genotypes in the qctool dosage format.** NPS expects individual genotype data are provided in the dosage file format. We use qctool to generate these files (See instructions). The genotype files need to be split by chromosomes for parallelization, and for each chromosome, the file should be named as "chrom*N.DatasetID*.dosage.gz."

3. **Sample IDs in the PLINK .fam format.** The samples in the .fam file should appear in the exactly same order as the samples in the genotype dosage files. This is a space- or tab-separated six-column text file without a column header. The phenotype information in this file will be ignored. See PLINK documentation for the details on the format.

```
trainF2  trainI2  0  0  0 -9
trainF3  trainI3  0  0  0 -9
trainF39 trainI39 0  0  0 -9
trainF41 trainI41 0  0  0 -9
trainF58 trainI58 0  0  0 -9
```

9. **Phenotypes in the PLINK phenotype format.** NPS looks up phenotypes in a separately prepared phenotype file. The phenotype name has to be "**Outcome**" with cases and controls encoded by **1** and **0**, respectively. **FID** and **IID** are used together to match samples to .fam file. This file is tab-delimited, and samples can appear in any order. Missing phenotypes (e.g. missing entry of samples in .fam file or phenotypes encoded by **-9**) are not allowed.

```
FID   IID    Outcome
trainF2  trainI2  0
trainF39 trainI39 0
trainF3  trainI3  1
trainF41 trainI41 1
trainF58 trainI58 0
```

Running NPS

Test cases

We provide two sets of simulated test cases. Due to their large file sizes, they are provided separately from the software distribution. Please download them from the Sunyaev Lab server (ftp://genetics.bwh.harvard.edu/download/schun/). Test set #1 is relatively small (225MB), and NPS can be run in less than ~30 mins in total on a modest desktop PC even without linear-algebra acceleration. In contrast, test set #2 is more realistic simulation (11GB) and will require serious computational resources. NPS will generate up to 1 TB of intermediate data and can take ~ 6 hours on computer clusters with linear-algebra acceleration.

Both simulated datasets were generated using our multivariate-normal simulator (See our NPS manuscript for the details). Briefly:

- **Test set #1.** The number of markers across the genome is limited to 100,449. We assume that all causal SNPs are included in those 100,449 SNPs. Note that this is an unrealistic assumption; with such a sparse SNP set, causal SNPs may not be necessarily genotyped or accurately tagged. The fraction of causal SNP is 0.005 (a total of 522 causal SNPs). The GWAS cohort size is 100,000. The training cohort has 2,500 cases and 2,500 controls. The validation cohort consists of 5,000 individuals without case over-sampling. The heritability is 0.5. The phenotype prevalence is 5%.

- **Test set #2.** The number of markers across the genome is 5,012,500. The fraction of causal SNP is 0.001 (a total of 5,008 causal SNPs). The GWAS cohort size is 100,000. The training cohort has 2,500 cases and 2,500 controls. The validation cohort consists of 5,000 samples without case over-sampling. The heritability is 0.5. The phenotype prevalence is 5%. This is one of the benchmark simulation datasets used in our manuscript, with 10-fold reduction in validation cohort size.

We assume that the test datasets will be downloaded and unpacked in the following directories:

```
cd nps-1.1.0/testdata/

tar -zxvf NPS.Test1.tar.gz
# This will create the following test data files in nps-1.1.0/testdata/Test1
# Test1/Test1.summstats.txt (PREFORMATTED GWAS summary statistics)
# Test1/chrom1.Test1.train.dosage.gz (training cohort genotypes)
# Test1/chrom2.Test1.train.dosage.gz (training cohort genotypes)
# ...
# Test1/Test1.train.2.5K_2.5K.fam (training cohort sample IDs)
# Test1/Test1.train.2.5K_2.5K.phen (training cohort phenotypes)
# Test1/chrom1.Test1.val.dosage.gz (validation cohort genotypes)
# Test1/chrom2.Test1.val.dosage.gz (validation cohort genotypes)
# ...
# Test1/Test1.val.5K.fam (validation cohort sample IDs)
# Test1/Test1.val.5K.phen (validation cohort phenotypes)

tar -zxvf NPS.Test2.tar.gz
# This will create the following test data in nps-1.1.0/testdata/Test2
# Test2/Test2.summstats.txt (PREFORMATTED GWAS summary statistics)
# Test2/chrom1.Test2.train.dosage.gz (training cohort genotypes)
# Test2/chrom2.Test2.train.dosage.gz (training cohort genotypes)
# ...
# Test2/Test2.train.2.5K_2.5K.fam (training cohort sample IDs)
# Test2/Test2.train.2.5K_2.5K.phen (training cohort phenotypes)
# Test2/chrom1.Test2.val.dosage.gz (validation cohort genotypes)
# Test2/chrom2.Test2.val.dosage.gz (validation cohort genotypes)
# ...
# Test2/Test2.val.5K.fam (validation cohort sample IDs)
# Test2/Test2.val.5K.phen (validation cohort phenotypes)
```

Running NPS on test set #1 without parallelization

NPS was designed with parallel processing on clusters in mind. For this, the algorithm is broken down into multiple steps, and computationally-intensive operations are split by chromosomes and run in parallel. For instructions on running it on computer clusters, move onto SGE and LSF sections after this.

For desktop computers, we provide a wrapper script (`run_all_chroms.sh`) to drive SGE cluster jobs sequentially, by processing one chromosome at a time. Test set #1 is a small dataset and can be tested on modest desktop computers without parallelization. The instructions below are based on the scenario that you will run the test set #1 on a desktop computer.

1. **Standardize genotypes.** The first step is to standardize the training genotypes to the mean of 0 and variance of 1 using `nps_stdgt.job`. The first parameter (`testdata/Test1`) is the location of training cohort data, where NPS will find chrom*N*.*DatasetID*.dosage.gz files. The second parameter (`Test1.train`) is the *DatasetID* of training cohort.

```
2. cd nps-1.1.0/
3.
   ./run_all_chroms.sh sge/nps_stdgt.job testdata/Test1 Test1.train
```

- Note: If you use NPS support scripts to harmonize training cohort data with summary statistics, *DatasetID* will be "*CohortName*.QC2" as the genotype files will be named as chrom*N*.*CohortName*.QC2.dosage.gz.

4. **Configure an NPS run.** Next, we run `npsR/nps_init.R` to configure an NPS run:

```
Rscript npsR/nps_init.R testdata/Test1/Test1.summstats.txt testdata/Test1
testdata/Test1/Test1.train.2.5K_2.5K.fam
testdata/Test1/Test1.train.2.5K_2.5K.phen Test1.train 80 testdata/Test1/npsdat
```

   The command arguments are:

   - GWAS summary statistics file in the PREFORMATTED format: `testdata/Test1/Test1.summstats.txt`
   - directory containing training genotypes: `testdata/Test1`
   - Sample information of training samples: `testdata/Test1/Test1.train.2.5K_2.5K.fam`
   - phenotypes of training samples: `testdata/Test1/Test1.train.2.5K_2.5K.phen`
   - DatasetID of training cohort genotypes: `Test1.train`
   - analysis window size: `80` SNPs. The window size of 80 SNPs for ~100,000 genome-wide SNPs is comparable to 4,000 SNPs for ~5,000,000 genome-wide SNPs.
   - directory to store NPS data: `testdata/Test1/npsdat`. The NPS configuration and all output files will be stored here.

   The set-up can be checked by running `nps_check.sh`. If `nps_check.sh` finds an error, `FAIL` message will be printed. If everything is fine, only `OK` messages will be reported:

```
./nps_check.sh testdata/Test1/npsdat/
```

```
NPS data directory: testdata/Test1/npsdat/
Verifying nps_init:
Checking testdata/Test1/npsdat//args.RDS ...OK (version 1.1)
Checking testdata/Test1/npsdat//log ...OK
Verifying nps_stdgt:
Checking testdata/Test1/chrom1.Test1.train ...OK
Checking testdata/Test1/chrom2.Test1.train ...OK
Checking testdata/Test1/chrom3.Test1.train ...OK
...
```

5. ** Separate out the GWAS-significant peaks as a separate partition. **

```
./run_all_chroms.sh sge/nps_gwassig.job testdata/Test1/npsdat/

# Check the results of last step
./nps_check.sh testdata/Test1/npsdat/
```

4. **Set up the decorrelated "eigenlocus" space.** This step sets up the decorrelated eigenlocus space by projecting the data into the decorrelated domain and pruning residual correlations bretween windows. This is one of the most time-consuming steps of NPS. The first argument to `nps_decor_prune.job` is the NPS data directory, in this case, `testdata/Test1/npsdat/`. The second argument is the window shift. We recommend running NPS four times on shifted windows and merging the results in the last step. Specifically, we recommend shifting analysis windows by 0, WINSZ * 1/4, WINSZ * 2/4 and WINSZ * 3/4 SNPs, where WINSZ is the size of analysis window. For test set #1, we use the WINSZ of 80, thus the recommended window shifts should be `0`, `20`, `40` and `60`.

```
5. ./run_all_chroms.sh sge/nps_decor_prune.job testdata/Test1/npsdat/ 0
6. ./run_all_chroms.sh sge/nps_decor_prune.job testdata/Test1/npsdat/ 20
7. ./run_all_chroms.sh sge/nps_decor_prune.job testdata/Test1/npsdat/ 40
8. ./run_all_chroms.sh sge/nps_decor_prune.job testdata/Test1/npsdat/ 60
9.
```

```
# Check the results of last step
./nps_check.sh testdata/Test1/npsdat/
```

11. **Partition the rest of data.** We define the partition scheme by running `npsR/nps_prep_part.R`. The first argument is the NPS data directory (`testdata/Test1/npsdat/`). The second and third arguments are the numbers of partitions. We recommend 10-by-10 double-partitioning on the intervals of eigenvalues of projection and estimated effect sizes in the eigenlocus space, thus last two arguments are `10` and `10`:

```
 Rscript npsR/nps_prep_part.R testdata/Test1/npsdat/ 10 10
```

Then, partitioned genetic risk scores will be calculated using training samples with `nps_part.job`. The first argument is the NPS data directory (`testdata/Test1/npsdat/`) and the second argument is the window shift (`0`, `20`, `40` or `60`):

```
./run_all_chroms.sh sge/nps_part.job testdata/Test1/npsdat/ 0
./run_all_chroms.sh sge/nps_part.job testdata/Test1/npsdat/ 20
./run_all_chroms.sh sge/nps_part.job testdata/Test1/npsdat/ 40
./run_all_chroms.sh sge/nps_part.job testdata/Test1/npsdat/ 60

# Check the results of last step
./nps_check.sh testdata/Test1/npsdat/
```

13. **Estimate per-partition shrinkage weights.** We estimate the per-partition weights using `npsR/nps_reweight.R`. The argument is the NPS data directory (`testdata/Test1/npsdat/`):

```
Rscript npsR/nps_reweight.R testdata/Test1/npsdat/
```

The re-weighted effect sizes should be converted back to the original per-SNP space from the eigenlocus space. This will store per-SNP re-weighted effect sizes in files of testdata/Test1/npsdat/Test1.train.win_*shift*.adjbetahat.chrom*N*.txt. The order of re-weighted effect sizes in these files are the same as the order of SNPs in the summary statistics file.

14. **Validate the accuracy of prediction model in a validation cohort.** Last, polygenic risk scores will be calculated for each chromosome and for each individual in the validation cohort using `sge/nps_score.dosage.job` as follows:

```
./run_all_chroms.sh sge/nps_score.dosage.job testdata/Test1/npsdat/
testdata/Test1/ Test1.val 0
./run_all_chroms.sh sge/nps_score.dosage.job testdata/Test1/npsdat/
testdata/Test1/ Test1.val 20
./run_all_chroms.sh sge/nps_score.dosage.job testdata/Test1/npsdat/
testdata/Test1/ Test1.val 40
./run_all_chroms.sh sge/nps_score.dosage.job testdata/Test1/npsdat/
testdata/Test1/ Test1.val 60

# Check the results
./nps_check.sh testdata/Test1/npsdat/
```

Here, the first argument for `sge/nps_score.dosage.job` is the NPS data directory (`testdata/Test1/npsdat/`), the second argument is the directory containing validation cohort data (`testdata/Test1/`), and the third argument is the DatasetID for validation genotypes. Since the genotype files for validation cohorts are named as chrom*N.Test1.val*.dosage.gz, DatasetID has to be `Test1.val`. The last argument is the window shift (`0`, `20`, `40` or `60`).

Finally, `npsR/nps_val.R` will combine polygenic risk scores across all shifted windows and report per-individual scores along with overall accuracy statistics:

```
Rscript npsR/nps_val.R testdata/Test1/npsdat/ Test1.val
testdata/Test1/Test1.val.5K.fam testdata/Test1/Test1.val.5K.phen
```

The command arguments are:

- NPS data directory: `testdata/Test1/npsdat/`
- DatasetID for validation genotypes: `Test1.val`
- sample IDs of validation cohort: `testdata/Test1/Test1.val.5K.fam`
- phenotypes of validation samples: `testdata/Test1/Test1.val.5K.phen`

`npsR/nps_val.R` will print out the following. Here, it reports the AUC of 0.8531 and Nagelkerke's R2 of 0.2693255 in the validation cohort. The polygenic risk score for each individuals in the cohort are stored in the file `testdata/Test1/Test1.val.5K.phen.nps_score`.

Producing a combined prediction model...OK (saved in testdata/Test1/Test1.val.5K.phen.nps_score) Observed-scale R2 = 0.1061336 Liability-scale R2 = 0.4693835 ... Data: 4729 controls < 271 cases. Area under the curve: 0.8776 95% CI: 0.8589-0.8963 (DeLong) Nagelkerke's R2 = 0.3172322

Running NPS on test set #1 using SGE clusters

To run NPS on SGE clusters, please run the following steps. All steps have to run in the top-level NPS directory (`nps-1.1.0/`), and jobs should be launched with the `qsub -cwd` option. The option `-t 1-22` will run NPS jobs over all 22 chromosomes in parallel. The job scripts are located in the `sge/` directory.

```
cd nps-1.1.0/

# Standardize genotypes
qsub -cwd -t 1-22 sge/nps_stdgt.job testdata/Test1 Test1.train

# Check the results
./nps_check.sh stdgt testdata/Test1 Test1.train

# Configure
Rscript npsR/nps_init.R testdata/Test1/Test1.summstats.txt testdata/Test1
testdata/Test1/Test1.train.2.5K_2.5K.fam testdata/Test1/Test1.train.2.5K_2.5K.phen
Test1.train 80 testdata/Test1/npsdat

# Check the results
./nps_check.sh init testdata/Test1/npsdat/

# Set up the decorrelated eigenlocus space
qsub -cwd -t 1-22 sge/nps_decor_prune_gwassig.job testdata/Test1/npsdat/ 0
qsub -cwd -t 1-22 sge/nps_decor_prune_gwassig.job testdata/Test1/npsdat/ 20
qsub -cwd -t 1-22 sge/nps_decor_prune_gwassig.job testdata/Test1/npsdat/ 40
qsub -cwd -t 1-22 sge/nps_decor_prune_gwassig.job testdata/Test1/npsdat/ 60

# Check the results of last step
./nps_check.sh last testdata/Test1/npsdat/ 0 20 40 60

# Define partitioning boundaries
Rscript npsR/nps_prep_part.R testdata/Test1/npsdat/ 0 10 10
Rscript npsR/nps_prep_part.R testdata/Test1/npsdat/ 20 10 10
Rscript npsR/nps_prep_part.R testdata/Test1/npsdat/ 40 10 10
Rscript npsR/nps_prep_part.R testdata/Test1/npsdat/ 60 10 10

# Calculate partitioned risk scores in the training cohort
qsub -cwd -t 1-22 sge/nps_part.job testdata/Test1/npsdat/ 0
qsub -cwd -t 1-22 sge/nps_part.job testdata/Test1/npsdat/ 20
qsub -cwd -t 1-22 sge/nps_part.job testdata/Test1/npsdat/ 40
qsub -cwd -t 1-22 sge/nps_part.job testdata/Test1/npsdat/ 60

# Check the results of last step
```

```
./nps_check.sh last testdata/Test1/npsdat/ 0 20 40 60

# Estimate per-partition shrinkage weights
Rscript npsR/nps_weight.R testdata/Test1/npsdat/ 0
Rscript npsR/nps_weight.R testdata/Test1/npsdat/ 20
Rscript npsR/nps_weight.R testdata/Test1/npsdat/ 40
Rscript npsR/nps_weight.R testdata/Test1/npsdat/ 60

# (Optional) Report the overall AUC of prediction in the training cohort
Rscript npsR/nps_train_AUC.R testdata/Test1/npsdat/ 0 20 40 60

# (Optional) Generate a plot of overall shrinkage curves
Rscript npsR/nps_plot_shrinkage.R testdata/Test1/npsdat/ Test1.nps.pdf 0 20 40 60

# Convert back to per-SNP effect sizes
qsub -cwd -t 1-22 sge/nps_back2snpeff.job testdata/Test1/npsdat/ 0
qsub -cwd -t 1-22 sge/nps_back2snpeff.job testdata/Test1/npsdat/ 20
qsub -cwd -t 1-22 sge/nps_back2snpeff.job testdata/Test1/npsdat/ 40
qsub -cwd -t 1-22 sge/nps_back2snpeff.job testdata/Test1/npsdat/ 60

# Check the results of last step
./nps_check.sh last testdata/Test1/npsdat/ 0 20 40 60

# Calculate polygenic scores for each chromosome and for each individual in the
validation cohort
qsub -cwd -t 1-22 sge/nps_score.job testdata/Test1/npsdat/ testdata/Test1/ Test1.val 0
qsub -cwd -t 1-22 sge/nps_score.job testdata/Test1/npsdat/ testdata/Test1/ Test1.val 20
qsub -cwd -t 1-22 sge/nps_score.job testdata/Test1/npsdat/ testdata/Test1/ Test1.val 40
qsub -cwd -t 1-22 sge/nps_score.job testdata/Test1/npsdat/ testdata/Test1/ Test1.val 60

# Check the results of nps_score
./nps_check.sh score testdata/Test1/npsdat/ testdata/Test1/ Test1.val 0 20 40 60

# Calculate overall polygenic scores and report prediction accuracies
Rscript npsR/nps_val.R testdata/Test1/npsdat/ testdata/Test1/
testdata/Test1/Test1.val.5K.fam testdata/Test1/Test1.val.5K.phen 0 20 40 60
```

Running NPS on test set #1 using LSF clusters

Running NPS on LSF clusters is similar. We provide the job scripts in `lsf/` directory.

```
cd nps-1.1.0/

# Standardize genotypes
bsub -J stdgt[1-22] lsf/nps_stdgt.job testdata/Test1 Test1.train

# Check the results
./nps_check.sh stdgt testdata/Test1 Test1.train

# Configure
Rscript npsR/nps_init.R testdata/Test1/Test1.summstats.txt testdata/Test1
testdata/Test1/Test1.train.2.5K_2.5K.fam testdata/Test1/Test1.train.2.5K_2.5K.phen
Test1.train 80 testdata/Test1/npsdat

# Check the results
./nps_check.sh init testdata/Test1/npsdat/

# Set up the decorrelate eigenlocus space
bsub -J decor[1-22] lsf/nps_decor_prune_gwassig.job testdata/Test1/npsdat/ 0
bsub -J decor[1-22] lsf/nps_decor_prune_gwassig.job testdata/Test1/npsdat/ 20
bsub -J decor[1-22] lsf/nps_decor_prune_gwassig.job testdata/Test1/npsdat/ 40
```

```
bsub -J decor[1-22] lsf/nps_decor_prune_gwassig.job testdata/Test1/npsdat/ 60

# Check the results of last step
./nps_check.sh last testdata/Test1/npsdat/ 0 20 40 60

# Define partitioning boundaries
Rscript npsR/nps_prep_part.R testdata/Test1/npsdat/ 0 10 10
Rscript npsR/nps_prep_part.R testdata/Test1/npsdat/ 20 10 10
Rscript npsR/nps_prep_part.R testdata/Test1/npsdat/ 40 10 10
Rscript npsR/nps_prep_part.R testdata/Test1/npsdat/ 60 10 10

# Calculate partitioned risk scores in the training cohort
bsub -J part[1-22] lsf/nps_part.job testdata/Test1/npsdat/ 0
bsub -J part[1-22] lsf/nps_part.job testdata/Test1/npsdat/ 20
bsub -J part[1-22] lsf/nps_part.job testdata/Test1/npsdat/ 40
bsub -J part[1-22] lsf/nps_part.job testdata/Test1/npsdat/ 60

# Check the results of last step
./nps_check.sh last testdata/Test1/npsdat/ 0 20 40 60

# Estimate per-partition shrinkage weights
Rscript npsR/nps_weight.R testdata/Test1/npsdat/ 0
Rscript npsR/nps_weight.R testdata/Test1/npsdat/ 20
Rscript npsR/nps_weight.R testdata/Test1/npsdat/ 40
Rscript npsR/nps_weight.R testdata/Test1/npsdat/ 60

# (Optional) Report the overall AUC of prediction in the training cohort
Rscript npsR/nps_train_AUC.R testdata/Test1/npsdat/ 0 20 40 60

# (Optional) Generate a plot of overall shrinkage curves
Rscript npsR/nps_plot_shrinkage.R testdata/Test1/npsdat/ Test1.nps.pdf 0 20 40 60

# Convert back to per-SNP effect sizes
bsub -J back2snpeff[1-22] lsf/nps_back2snpeff.job testdata/Test1/npsdat/ 0
bsub -J back2snpeff[1-22] lsf/nps_back2snpeff.job testdata/Test1/npsdat/ 20
bsub -J back2snpeff[1-22] lsf/nps_back2snpeff.job testdata/Test1/npsdat/ 40
bsub -J back2snpeff[1-22] lsf/nps_back2snpeff.job testdata/Test1/npsdat/ 60

# Check the results of last step
./nps_check.sh last testdata/Test1/npsdat/ 0 20 40 60

# Calculate polygenic scores for each chromosome and for each individual in the
validation cohort
bsub -J score[1-22] lsf/nps_score.job testdata/Test1/npsdat/ testdata/Test1/ Test1.val
0
bsub -J score[1-22] lsf/nps_score.job testdata/Test1/npsdat/ testdata/Test1/ Test1.val
20
bsub -J score[1-22] lsf/nps_score.job testdata/Test1/npsdat/ testdata/Test1/ Test1.val
40
bsub -J score[1-22] lsf/nps_score.job testdata/Test1/npsdat/ testdata/Test1/ Test1.val
60

# Check the results of nps_score
./nps_check.sh score testdata/Test1/npsdat/ testdata/Test1/ Test1.val 0 20 40 60

# Calculate overall polygenic scores and report prediction accuracies
Rscript npsR/nps_val.R testdata/Test1/npsdat/ testdata/Test1/
testdata/Test1/Test1.val.5K.fam testdata/Test1/Test1.val.5K.phen 0 20 40 60
```

Running NPS on test set #2 using SGE clusters

NPS can be run on test set #2 similarly as test set #1 except:

- Since test set #2 has a total of ~5,000,000 genome-wide common SNPs, we recommend to use the window size of **4,000 SNPs**. And accordingly, window shifts should be set to **0, 1,000, 2,000, and 3,000 SNPs**. This is the setting we generally recommend for real data as well.
- Some of NPS steps now require large memory space. With most datasets, 4GB memory space per a task is sufficient for NPS. On SGE, the memory requirement can be specified by `qsub -l h_vmem=4G`.
- For test set #2 and real data sets, we do not recommend running NPS without parallelization because of heavy computational requirements.

```
cd nps-1.1.0/

# Standardize genotypes
qsub -cwd -t 1-22 sge/nps_stdgt.job testdata/Test2/ Test2.train

# Check the results
./nps_check.sh stdgt testdata/Test2/ Test2.train

# Configure
# CAUTION: This step requires large memory space
# You may need to run this as a job or open an interative session for it by:
# qlogin -l h_vmem=4G
Rscript npsR/nps_init.R testdata/Test2/Test2.summstats.txt testdata/Test2
testdata/Test2/Test2.train.2.5K_2.5K.fam testdata/Test2/Test2.train.2.5K_2.5K.phen
Test2.train 4000 testdata/Test2/npsdat

# Check the results
./nps_check.sh init testdata/Test2/npsdat/

# Set up the decorrelate eigenlocus space
qsub -cwd -l h_vmem=4G -t 1-22 sge/nps_decor_prune_gwassig.job testdata/Test2/npsdat/ 0
qsub -cwd -l h_vmem=4G -t 1-22 sge/nps_decor_prune_gwassig.job testdata/Test2/npsdat/
1000
qsub -cwd -l h_vmem=4G -t 1-22 sge/nps_decor_prune_gwassig.job testdata/Test2/npsdat/
2000
qsub -cwd -l h_vmem=4G -t 1-22 sge/nps_decor_prune_gwassig.job testdata/Test2/npsdat/
3000

# Check the results of last step
./nps_check.sh last testdata/Test2/npsdat/ 0 1000 2000 3000

# Define partitioning boundaries
Rscript npsR/nps_prep_part.R testdata/Test2/npsdat/ 0 10 10
Rscript npsR/nps_prep_part.R testdata/Test2/npsdat/ 1000 10 10
Rscript npsR/nps_prep_part.R testdata/Test2/npsdat/ 2000 10 10
Rscript npsR/nps_prep_part.R testdata/Test2/npsdat/ 3000 10 10

# Calculate partitioned risk scores in the training cohort
qsub -cwd -l h_vmem=4G -t 1-22 sge/nps_part.job testdata/Test2/npsdat/ 0
qsub -cwd -l h_vmem=4G -t 1-22 sge/nps_part.job testdata/Test2/npsdat/ 1000
qsub -cwd -l h_vmem=4G -t 1-22 sge/nps_part.job testdata/Test2/npsdat/ 2000
qsub -cwd -l h_vmem=4G -t 1-22 sge/nps_part.job testdata/Test2/npsdat/ 3000

# Check the results of last step
./nps_check.sh last testdata/Test2/npsdat/ 0 1000 2000 3000

# Estimate per-partition shrinkage weights
```

```
Rscript npsR/nps_weight.R testdata/Test2/npsdat/ 0
Rscript npsR/nps_weight.R testdata/Test2/npsdat/ 1000
Rscript npsR/nps_weight.R testdata/Test2/npsdat/ 2000
Rscript npsR/nps_weight.R testdata/Test2/npsdat/ 3000

# (Optional) Report the overall AUC of prediction in the training cohort
Rscript npsR/nps_plot_shrinkage.R testdata/Test2/npsdat/ Test2.nps.pdf 0 1000 2000 3000

# (Optional) Generate a plot of overall shrinkage curves
Rscript npsR/nps_train_AUC.R testdata/Test2/npsdat/ 0 1000 2000 3000

qsub -cwd -l h_vmem=4G -t 1-22 sge/nps_back2snpeff.job testdata/Test2/npsdat/ 0
qsub -cwd -l h_vmem=4G -t 1-22 sge/nps_back2snpeff.job testdata/Test2/npsdat/ 1000
qsub -cwd -l h_vmem=4G -t 1-22 sge/nps_back2snpeff.job testdata/Test2/npsdat/ 2000
qsub -cwd -l h_vmem=4G -t 1-22 sge/nps_back2snpeff.job testdata/Test2/npsdat/ 3000

# Check the results of last step
./nps_check.sh last testdata/Test2/npsdat/ 0 1000 2000 3000

# Convert back to per-SNP effect sizes
qsub -cwd -t 1-22 sge/nps_score.job testdata/Test2/npsdat/ testdata/Test2/ Test2.val 0
qsub -cwd -t 1-22 sge/nps_score.job testdata/Test2/npsdat/ testdata/Test2/ Test2.val
1000
qsub -cwd -t 1-22 sge/nps_score.job testdata/Test2/npsdat/ testdata/Test2/ Test2.val
2000
qsub -cwd -t 1-22 sge/nps_score.job testdata/Test2/npsdat/ testdata/Test2/ Test2.val
3000

# Check the results of nps_score
./nps_check.sh score testdata/Test2/npsdat/ testdata/Test2/ Test2.val 0 1000 2000 3000

# Calculate polygenic scores for each chromosome and for each individual in the
validation cohort
Rscript npsR/nps_val.R testdata/Test2/npsdat/ testdata/Test2/
testdata/Test2/Test2.val.5K.fam testdata/Test2/Test2.val.5K.phen 0 1000 2000 3000
```

`nps_train_AUC.R` will report the following AUC in the training cohort:

Data: 2500 controls < 2500 cases.
Area under the curve: **0.7843**
95% CI: 0.7718-0.7968 (DeLong)

`nps_plot_shrinkage.R` will plot the curves of estimated conditional mean effects and save it to a pdf file (`Test2.nps.pdf`).

`nps_val.R` will report the following overall prediction accuracy in the validation cohort:

Non-Parametric Shrinkage 1.1
Validation cohort:
Total 5000 samples
240 case samples
4760 control samples
0 samples with missing phenotype (-9)
Includes TotalLiability
Checking a prediction model (winshift = 0)...
Observed-scale R2 = 0.04862955
Liability-scale R2 = 0.2303062
Checking a prediction model (winshift = 1000)...
Observed-scale R2 = 0.04994584
Liability-scale R2 = 0.2298484
Checking a prediction model (winshift = 2000)...
Observed-scale R2 = 0.05150205
Liability-scale R2 = 0.2268046

Checking a prediction model (winshift = 3000)...
Observed-scale R2 = 0.05258402
Liability-scale R2 = 0.2298871

Producing a combined prediction model...OK (saved
in **testdata/Test2/Test2.val.5K.phen.nps_score**)
Observed-scale R2 = 0.05253146
Liability-scale R2 = 0.2376991
Loading required package: pROC
Type 'citation("pROC")' for a citation.

Attaching package: ‘pROC’

The following objects are masked from ‘package:stats’:

cov, smooth, var

AUC:

Call:
roc.default(controls = prisk[vlY == 0], cases = prisk[vlY == 1], ci = TRUE)

Data: 4760 controls < 240 cases.
Area under the curve: 0.7886
95% CI: 0.7617-0.8154 (DeLong)
Loading required package: DescTools
Nagelkerke's R2 = 0.1668188

Call: glm(formula = vlY ~ prisk, family = binomial(link = "logit"))

Coefficients:
(Intercept)   prisk
    -5.2888   0.2387

Degrees of Freedom: 4999 Total (i.e. Null); 4998 Residual
Null Deviance: 1926
Residual Deviance: 1652   AIC: 1656
Done

Running NPS on test set #2 using LSF clusters

Running NPS on LSF is similar to running it on SGE clusters. The memory limit is specified by `bsub -R 'rusage[mem=4000]'`.

```
cd nps-1.1.0/

# Standardize genotypes
bsub -J stdgt[1-22] lsf/nps_stdgt.job testdata/Test2/ Test2.train

# Check the results
./nps_check.sh stdgt testdata/Test2/ Test2.train

# Configure
# This step requires large memory space.
# You may need to run this as a job or open an interative session for it by running:
# bsub -Is -R 'rusage[mem=4000]' /bin/bash
```

```
Rscript npsR/nps_init.R testdata/Test2/Test2.summstats.txt testdata/Test2
testdata/Test2/Test2.train.2.5K_2.5K.fam testdata/Test2/Test2.train.2.5K_2.5K.phen
Test2.train 4000 testdata/Test2/npsdat

# Check the results
./nps_check.sh init testdata/Test2/npsdat/

# Set up the decorrelate eigenlocus space
bsub -R 'rusage[mem=4000]' -J decor[1-22] lsf/nps_decor_prune_gwassig.job
testdata/Test2/npsdat/ 0
bsub -R 'rusage[mem=4000]' -J decor[1-22] lsf/nps_decor_prune_gwassig.job
testdata/Test2/npsdat/ 1000
bsub -R 'rusage[mem=4000]' -J decor[1-22] lsf/nps_decor_prune_gwassig.job
testdata/Test2/npsdat/ 2000
bsub -R 'rusage[mem=4000]' -J decor[1-22] lsf/nps_decor_prune_gwassig.job
testdata/Test2/npsdat/ 3000

# Check the results of last step
./nps_check.sh last testdata/Test2/npsdat/ 0 1000 2000 3000

# Define partitioning boundaries
Rscript npsR/nps_prep_part.R testdata/Test2/npsdat/ 0 10 10
Rscript npsR/nps_prep_part.R testdata/Test2/npsdat/ 1000 10 10
Rscript npsR/nps_prep_part.R testdata/Test2/npsdat/ 2000 10 10
Rscript npsR/nps_prep_part.R testdata/Test2/npsdat/ 3000 10 10

# Calculate partitioned risk scores in the training cohort
bsub -R 'rusage[mem=4000]' -J part[1-22] lsf/nps_part.job testdata/Test2/npsdat/ 0
bsub -R 'rusage[mem=4000]' -J part[1-22] lsf/nps_part.job testdata/Test2/npsdat/ 1000
bsub -R 'rusage[mem=4000]' -J part[1-22] lsf/nps_part.job testdata/Test2/npsdat/ 2000
bsub -R 'rusage[mem=4000]' -J part[1-22] lsf/nps_part.job testdata/Test2/npsdat/ 3000

# Check the results of last step
./nps_check.sh last testdata/Test2/npsdat/ 0 1000 2000 3000

# Estimate per-partition shrinkage weights
Rscript npsR/nps_weight.R testdata/Test2/npsdat/ 0
Rscript npsR/nps_weight.R testdata/Test2/npsdat/ 1000
Rscript npsR/nps_weight.R testdata/Test2/npsdat/ 2000
Rscript npsR/nps_weight.R testdata/Test2/npsdat/ 3000

# (Optional) Report the overall AUC of prediction in the training cohort
Rscript npsR/nps_plot_shrinkage.R testdata/Test2/npsdat/ Test2.nps.pdf 0 1000 2000 3000

# (Optional) Generate a plot of overall shrinkage curves
Rscript npsR/nps_train_AUC.R testdata/Test2/npsdat/ 0 1000 2000 3000

# Convert back to per-SNP effect sizes
bsub -R 'rusage[mem=4000]' -J back2snpeff[1-22] lsf/nps_back2snpeff.job
testdata/Test2/npsdat/ 0
bsub -R 'rusage[mem=4000]' -J back2snpeff[1-22] lsf/nps_back2snpeff.job
testdata/Test2/npsdat/ 1000
bsub -R 'rusage[mem=4000]' -J back2snpeff[1-22] lsf/nps_back2snpeff.job
testdata/Test2/npsdat/ 2000
bsub -R 'rusage[mem=4000]' -J back2snpeff[1-22] lsf/nps_back2snpeff.job
testdata/Test2/npsdat/ 3000

# Check the results of last step
./nps_check.sh last testdata/Test2/npsdat/ 0 1000 2000 3000
```

```
# Calculate polygenic scores for each chromosome and for each individual in the
validation cohort
bsub -J score[1-22] lsf/nps_score.job testdata/Test2/npsdat/ testdata/Test2/ Test2.val
0
bsub -J score[1-22] lsf/nps_score.job testdata/Test2/npsdat/ testdata/Test2/ Test2.val
1000
bsub -J score[1-22] lsf/nps_score.job testdata/Test2/npsdat/ testdata/Test2/ Test2.val
2000
bsub -J score[1-22] lsf/nps_score.job testdata/Test2/npsdat/ testdata/Test2/ Test2.val
3000

# Check the results of nps_score
./nps_check.sh score testdata/Test2/npsdat/ testdata/Test2/ Test2.val 0 1000 2000 3000

# Calculate polygenic scores for each chromosome and for each individual in the
validation cohort
Rscript npsR/nps_val.R testdata/Test2/npsdat/ testdata/Test2/
testdata/Test2/Test2.val.5K.fam testdata/Test2/Test2.val.5K.phen 0 1000 2000 3000
```

How to prepare training and validation cohorts for NPS

We take an example of UK Biobank to show how to prepare training and validation cohorts for NPS. In principle, however, NPS can work with other cohorts as far as the genotype data are prepared in the bgen file format. To gain access to UK Biobank, please check UK Biobank data access application procedure.

Using UK Biobank as a training cohort

UK Biobank data consist of the following files:

- **ukb_imp_chrN_v3.bgen**: imputed allelic dosages by chromosomes
- **ukb_mfi_chrN_v3.txt**: marker information by chromosomes
- **ukb31063.sample**: bgen sample information file for the entire cohort

Assuming that UK Biobank dataset is located in `<path_to_ukbb>/`, we first exclude all Indels and SNPs with minor allele frequency < 5% or imputation quality (INFO) score < 0.4 by running the following:

```
qsub -l h_vmem=4G -t 1-22 support/common_snps.job <path_to_ukbb>/ukb_imp_chr#_v3.bgen
<path_to_ukbb>/ukb_mfi_chr#_v3.txt <work_dir>
```

The command arguments are:

- file path to bgen files: `<path_to_ukbb>/ukb_imp_chr#_v3.bgen` with chromosome numbers replacing #
- file path to marker information files: `<path_to_ukbb>/ukb_mfi_chr#_v3.txt` with chromosome numbers replacing #
- work directory: `<work_dir>`, where output files will be saved.

Next, we filter bgen files to include only training cohort samples (as specified in `<sample_id_file>`) and then export the filtered genotypes into dosage files.
The `<sample_id_file>` is simply a list of sample IDs, with one sample in each line. This can be done as follows:

```
qsub -l h_vmem=4G -t 1-22 support/filter_samples.job <path_to_ukbb>/ukb31063.sample
<work_dir> <sample_id_file> <training_cohort_name>
```

The command arguments are:

- bgen sample file for the entire cohort: `<path_to_ukbb>/ukb31063.sample`
- work directory: `<work_dir>`, where output files will be saved
- file listing the sample IDs for training cohort: `<sample_id_file>`
- name of the training cohort: `<training_cohort_name>`. Name should not contain a whitespace character. Output files will be named after `<training_cohort_name>`.

Then, we harmonize GWAS summary statitics with training cohort data:

```
# CAUTION: This script uses large memory space.
Rscript support/harmonize_summstats.R <summary_statistics_file> <work_dir>
<training_cohort_name>
```

The command arguments are:

- GWAS summary statistics: `<summary_statistics_file>` in the *MINIMAL* format
- work directory: `<work_dir>`, where output files will be saved
- name of the training cohort: `<training_cohort_name>` used in the previous step with `support/filter_samples.job`

`support/harmonize_summstats.R` will run QC filters and generate the harmonized GWAS summary statistics in the *PREFORMATTED* format (`<work_dir>/<training_cohort_name>.preformatted_summstats.txt`), which can be now used for core NPS modules. Specifically, the following QCs measures will be taken:

- check missing values or numerical underflows in summary statistics
- remove tri-allelic SNPs
- assign reference and alternative alleles
- remove duplicated markers
- restrict to the markers overlapping between GWAS and training data
- cross-check allele frequencies between datasets if **effaf** is provided in the summary statistics file. Variants with too discordant allele frequencies will be rejected to prevent potential allele flips.

After that, we need to filter out SNPs in the training genotype files that were flagged for removal during the above harmonization step as follows:

```
qsub -l h_vmem=4G -t 1-22 support/filter_variants.job <work_dir>
<training_cohort_name>
```

Finally, the following step will create `<work_dir>/<training_cohort_name>.QC2.fam`, which keeps tracks of the IDs of all training cohort samples:

```
support/make_fam.sh <work_dir> <training_cohort_name>.QC2
```

Here, we extract the sample IDs from the column header of training genotype dosage file and fill **both FID and IID** of .fam file with the same sample IDs. *If this behavior is not desirable, the .fam file has to be manually created.*

Overall, the job scripts will automatially generate the following set of NPS input files:

- `<work_dir>/<training_cohort_name>.preformatted_summstats.txt`
- `<work_dir>/chromN.<training_cohort_name>.QC2.dosage.gz`
- `<work_dir>/<training_cohort_name>.QC2.fam`

Note:

- `common_snps.job` and `filter_samples.job` use bgen and qctool, respectively. The job scripts may need to be moditifed to load these modules.
- The job scripts in `support/` directory is written for SGE clusters but can be easily ported to LSF or other cluster systems.
- The job scripts use memory space up to 4GB (run with `qsub -l h_vmem=4G`). Depending on data,`support/harmonize_summstats.R` can take memory up to ~8GB. It will terminate abruptly if it runs out of memory.

Using a different cohort as a training cohort

To use other cohort as a training cohort, you will need to generate marker information files similar to **ukb_mfi_chrN_v3.txt** of UK Biobank. We can generate these files from bgen files (**<dataset_dir>/chromN.bgen**) as follows:

```
qsub -t 1-22 support/make_snp_info.job <dataset_dir>/chrom#.bgen
<work_dir>
```

The first parameter (`<dataset_dir>/chrom#.bgen`) is the path to bgen genotype files, with # replacing a chromosome number. Internally, `support/make_snp_info.job` relies on **qctool**, thus the job script may need to be modified to load the module if needed. The output marker information files will be saved in `<work_dir>` and named as "**chrom*N*.mfi.txt**".
The rest of steps are straight-forward and similar to using UK Biobank data:

```
# Filter out InDels and SNPs with MAF < 5% or INFO < 0.4
qsub -l h_vmem=4G -t 1-22 support/common_snps.job <dataset_dir>/chrom#.bgen
<work_dir>/chrom#.mfi.txt <work_dir>

# Restrict samples to those specified in <sample_id_file> and export .dosage.gz files
qsub -l h_vmem=4G -t 1-22 support/filter_samples.job
<bgen_sample_file_of_entire_cohort> <work_dir> <sample_id_file> <training_cohort_name>

# Harmonize GWAS summary statistics with training genotype data
Rscript support/harmonize_summstats.R <summary_statistics_file> <work_dir>
<training_cohort_name>

# Run extra variant filtering
qsub -l h_vmem=4G -t 1-22 support/filter_variants.job <work_dir> <training_cohort_name>

# Generate .fam file with identical IIDs and FIDs
support/make_fam.sh <work_dir> <training_cohort_name>.QC2
```

Using UK Biobank for a validation as well as a training cohort

UK Biobank can be split into two and used as a validation as well as a training cohort. Assume that `<sample_id_file>` contains the IDs of samples to include in the validation cohort. Then, validation cohort can be prepared for NPS similarly:

```
# import the list of SNPs rejected while harmonizing the training cohort into the
validation cohort
cp <work_dir>/<training_cohort_name>.UKBB_rejected_SNPIDs
<work_dir>/<validation_cohort_name>.UKBB_rejected_SNPIDs

# Restrict samples to those specified in <sample_id_file> and export .dosage.gz files
qsub -t 1-22 support/filter_samples.job <path_to_ukbb>/ukb31063.sample <work_dir>
<sample_id_file> <validation_cohort_name>
```

```
# Run extra variant filtering with
<work_dir>/<validation_cohort_name>.UKBB_rejected_SNPIDs
qsub -t 1-22 support/filter_variants.job <work_dir> <validation_cohort_name>

# Generate .fam file with identical IIDs and FIDs
support/make_fam.sh <work_dir> <validation_cohort_name>.QC2
```

The `<training_cohort_name>.UKBB_rejected_SNPIDs` file contains the list of SNPs that were rejected while harmonizing the GWAS summary statistics with training cohort data. This file has to be copied to the validation cohort so that training and validation cohorts will have the same set of markers after running `support/filter_variants.job`.

Using a validation cohort that is independent from a training cohort

To help deploying NPS polygenic scores to a cohort that is independent from a training cohort, we provide `sge/nps_harmonize_val.job` (`lsf/nps_harmonize_val.job` for LSF). This will convert bgen files into the dosage file format and perform the following:

- Remove SNPs that were not defined in the trained polygenic score model.
- Fill in SNPs that are missing in the validation cohort with 0.
- Make sure that the validation cohort genotype files and polygenic model have the consistent ordering of markers.

This will be done by running `nps_harmonize_val.job` as follows:

```
# Generate <work_dir>/chromN.<cohort_name>.dosage.gz files
qsub -t 1-22 -l h_vmem=4G sge/nps_harmonize_val.job <nps_data_dir>
<dataset_dir>/chrom#.bgen <bgen_sample_file> <work_dir> <cohort_name>

# Generate <work_dir>/<cohort_name>.fam file with identical IIDs and FIDs
support/make_fam.sh <work_dir> <cohort_name>
```

The command arguments are:

- NPS data directory: `<nps_data_dir>` to locate the marker information of trained polygenic risk score
- file path to bgen files of validation cohort: `<dataset_dir>/chrom#.bgen` with chromosome number replacing #.
- bgen sample file: `<bgen_sample_file>` contains the sample information of cohort
- work directory: `<work_dir>`, where output files will be saved.

Note:

- `nps_harmonize_val.job` relies on **qctool** internally. The job script may need to be modified to load the module.
- This script will generate harmonized genotype files named as "chrom*N.<cohort_name>*.dosage.gz". DatasetID to designate this files in NPS will be just *<cohort_name>*.
- Currently, `support/make_fam.sh` produces .fam files with identical FID and IID, which are extracted from .dosage.gz files.