## Installation

The programs are mostly written in Python, except for a few C++ and Cython code.

It is recommended to install Anaconda (version 4.4.0, can be downloaded from (https://repo.continuum.io/archive/Anaconda2-4.4.0-Linux-x86\_64.sh)) and the following Python packages:

pip install h5py tqdm

Only one C++ program genotype\_to\_corpus needs to be compiled:

make

and the program binary will be generated in bin/genotype\_to\_corpus.

The Python scripts are in the bin/ directory.

## Instructions

### Input files

Assume that the input raw data is in a directory named emaize\_data, and the directory structure is like:

emaize\_data/  
├── genotype  
│   ├── chr10\_emaize.genoMat  
│   ├── chr1\_emaize.genoMat  
│   ├── chr2\_emaize.genoMat  
│   ├── chr3\_emaize.genoMat  
│   ├── chr4\_emaize.genoMat  
│   ├── chr5\_emaize.genoMat  
│   ├── chr6\_emaize.genoMat  
│   ├── chr7\_emaize.genoMat  
│   ├── chr8\_emaize.genoMat  
│   └── chr9\_emaize.genoMat  
├── phenotype  
│   ├── pheno\_emaize.txt  
│   └── phenotype\_fm\_table.txt  
└── README\_eMaize\_data\_eng.pdf

The first 4 lines of emaize\_data/genotype is like:

snp alleles chrom posi L0001 L0002 L0003 L0004 L0005 L0006 L0007 L0008 L0009 L0010  
chr1.s\_5402 A/T 1 5402 AA AA AA AT AA AA AA AA AA  
chr1.s\_6490 A/G 1 6490 AA AA AA AA AA AA AA AA AA  
chr1.s\_6707 T/G 1 6707 TT TT TT TG TT TT TT TT TT

### Extract the sample names

[ -d data ] || mkdir data  
head -n 1 emaize\_data/genotype/chr1\_emaize.genoMat \  
 | tr '\t' '\n' | sed '1,4 d' > data/sample\_names.txt

### Convert genotype data to binary format (3-bit code, samples first)

Assume that a genotype has two alleles: A and B, then a genotype is converted to 3-bit code following the rules: AA -> 100, AB -> 010, BB -> 001.

[ -d data/genotype ] || mkdir -p data/genotype  
for i in $(seq 1 10);do  
 cat emaize\_data/genotype/chr${i}\_emaize.genoMat | bin/genotype\_to\_corpus > data/genotype/chr${i}  
done

This will create a directory data/genotype, and generate a binary file named data/genotype/$chrom for each chromosome. Each output file data/genotype/$chrom is the dump of a 2D C array of shape (n\_features\*3, n\_samples) with dtype = int8.

### Convert genotypes from 3-bit code to 2-bit code

Assume that a genotype has two alleles: A and B, then a genotype is converted to 2-bit code following the rules: AA -> 10, AB -> 11, BB -> 01.

for chrom in $(ls data/genotype);do  
 bin/convert\_3bit\_to\_2bit.py -i data/genotype/$chrom \  
 --phenotype-file emaize\_data/phenotype/pheno\_emaize.txt \  
 -o data/genotype\_2bit/$chrom  
done

This will create a directory data/genotype\_2bit, and generate an output file data/genotype\_2bit/$chrom for each chromosome. Each output file data/genotype\_2bit/$chrom is in HDF5 format.

### Convert 2-bit code to minor allele copy numbers

Assume that a genotype has a major allele A and a minor allele B, then a genotype is converted to an integer using the copy number of the major allele: AA -> 2, AB -> 1, BB -> 0. An allele is considered as major allele if the frequency of the allele frequency among all samples is larger than 50%.

for chrom in $(ls data/genotype);do  
 bin/normalize\_genotypes.py 2bit\_to\_minor -i data/genotype\_2bit/${chrom}:data -o data/genotype\_minor/$chrom  
done

This will create a directory data/genotype\_minor, and generate an output file data/genotype\_minor/$chrom for each chromosome. Each output file data/genotype\_2bit/$chrom is in HDF5 format.

### Extract genomic positions of selected SNPs

bin/preprocess.py extract\_snp\_pos -i emaize\_data/genotype -o data/genomic\_positions

This will create an output file data/genomic\_positions.

### Sample SNPs from the whole genome uniformly (10000, 100000)

for n\_snps in 10000 100000;do  
 bin/preprocess.py random\_select -i data/genotype\_minor \  
 --genomic-pos-file data/genomic\_positions -n $n\_snps -k 1 -o output/random\_select/$n\_snps  
done

This will create two files output/random\_select/10000 and output/random\_select/100000. Each output file is an HDF5 file with k groups and each group contains 3 datasets: /$group/X, /$group/chrom, /$group/positions.

### Calculate genetic similarity matrix

bin/preprocess.py create\_gsm -i output/random\_select/100000:/0/X \  
 -o output/gsm/random\_select/100000/0

This will create an output file output/gsm/random\_select/100000/0. Each output file contains 4 datasets: K (genetic similarity matrix), U, S, V (result of SVD on K).

### Generate parent table from phenotype file

bin/preprocess.py generate\_parent\_table \  
 -i emaize\_data/phenotype/pheno\_emaize.txt \  
 -o data/parent\_table

This will create an output file data/parent\_table, which contains one dataset data. The dataset is a matrix with 30 rows (male parents) and 207 columns (female parents).

### Convert phenotypes from plain text to HDF5 format

Assume that the phenotypes are in file emaize\_data/phenotype/pheno\_emaize.txt with the first few lines:

type id pedigree trait1 trait2 trait3  
training L0001 f1\_X\_m1 -1.74610282478836 -0.785525121573821 -0.331636965445395  
training L0002 f2\_X\_m1 -1.67924837319023 -1.5694898694515 -2.57261413504835  
training L0003 f3\_X\_m1 -2.74891959876045 -0.608643883224438 -1.10881183080429  
training L0004 f4\_X\_m1 -2.41464734076976 -0.672045141494308 -1.31505078714756  
training L0005 f5\_X\_m1 -1.87981172798464 -0.740912762992952 -1.87918131205325

Run the following command to convert the table to HDF5 formatL

bin/preprocess.py phenotypes\_to\_hdf5 -i emaize\_data/phenotype/pheno\_emaize.txt \  
 -o data/phenotypes/all

This will create an output file data/phenotypes/all in HDF5 format with 6 datasets that correpond to the 6 columns in the file.

### Convert training and test indices to HDF5 format

Read the phenotype file emaize\_data/phenotype/pheno\_emaize.txt and extract the sample indices with the 'type' column specified as 'training' or 'test'.

bin/preprocess.py phenotypes\_to\_train\_test\_indices -i emaize\_data/phenotype/pheno\_emaize.txt \  
 -o data/train\_test\_indices

This will create an output file data/train\_test\_indices that contains two datasets: train, test.

### Select a subset (200 or 300 SNPs) from 10000 SNPs

for n\_snps in 100 200 300;do  
 bin/preprocess.py random\_select\_subset -i output/random\_select/10000 \  
 -m random\_choice -n $n\_snps --n-groups 200 -o output/random\_select\_subset/10000/random\_choice/${n\_snps}  
done

This will create an output file output/random\_select\_subset/10000/random\_choice/${n\_snps} for each SNP set size.

## Training and test

### Convert sample indices of training set and test set to HDF5 format

First prepare two text files named $train\_index\_file and $test\_index\_file that contain the indices of the training and test samples, one per line. And run:

bin/preprocess.py convert\_train\_test\_indices \  
 --train-index-file $train\_index\_file \  
 --test-index-file $test\_index\_file \  
 -o output/train\_test\_indices/0

This will create an output file output/train\_test\_indices/0 in HDF5 format with two datasets: train, test.

### Mixed ridge on a subset of 10000 SNPs (random choice)

for gamma in 0.05 0.10 0.15 0.20;do  
 for n\_snps in 200 300;do  
 for snp\_set in $(seq 0 199);do  
 for trait in $traits;do  
 bin/run\_mixed\_model.py mixed\_ridge \  
 --genotype-file output/random\_select\_subset/10000/random\_choice/${n\_snps}:/$snp\_set/X \  
 --transpose-genotype \  
 --gsm-file output/gsm/random\_select/100000/1 \  
 --phenotype-file data/phenotypes/all:$trait \  
 --parent-table-file data/parent\_table \  
 --train-index-file data/train\_test\_indices:/train \  
 --test-index-file data/train\_test\_indices:/test \  
 --cv-type $cv\_type \  
 --gammas $gamma --alphas 0.001 \  
 -o output/mixed\_ridge/10000/random\_choice/gamma=${gamma}/${n\_snps}/$trait/$snp\_set/$cv\_type  
 done  
 done  
 done  
done

### Select best subset of SNPs based on CV MSE

bin/run\_mixed\_model.py select\_best\_subset \  
 -i output/mixed\_ridge/10000/random\_choice \  
 --genotype-dir --genotype-file output/random\_select\_subset/10000/random\_choice \  
 --test-index-file data/train\_test\_indices:/test \  
 --gammas 0.05,0.10,0.15,0.20 --n-snps 200,300 --n-groups 200 \  
 --by mse\_cv\_mean \  
 -o output/select\_best\_subset/10000/random\_choice

This will generate two summary tables: \* output/select\_best\_subset/10000/summary.txt: cross-validation results of all parameter combinations and SNP subsets. \* output/select\_best\_subset/10000/summary\_best.txt: results of best SNP subsets that are selected with cross-validation. Each line is a combination of SNP subset and parameter combination. The script will also generate the best predictions on the test samples: output/select\_best\_subset/10000/${trait}.${rank}.txt.

## Reproduce the final predictions

Several files are required to reproduce the final predictions: \* best\_subsets/snps: selected SNP subsets. \* output/gsm/random\_select/100000/1: genetic similarity matrix and results of SVD. \* data/phenotypes/all: phenotypes of the training samples. \* data/parent\_table: a 2D table with rows as male parents and columns as female parents. \* data/train\_test\_indices: indices of training samples and test samples. Run

./bin/reproduce\_final\_predictions.sh

This will create a text file best\_subsets/predictions.txt.