README

March 25, 2020

1 Contact: Chao Ning

ningchao(at)sdau(dot)edu(dot)cn
ningchao91(at)gmail(dot)com

2 Install

GMAT will keep updating. Please uninstall older version to obtain the latest functions. The easiest uninstall way:
> pip uninstall gmat

2.1 Dependencies

- numpy>=1.16.0
- pandas>=0.19.0
- scipy>=1.1.1
- cffi>=1.12.0
- pandas_plink>=2.0.0
- tqdm>=4.43.0

We recommend using a Python distribution such as Anaconda (Python 3.7 version). This distribution can be used on Linux and Windows and is free. It is the easiest way to get all the required package dependencies.

2.2 Quick install

> pip install gmat

2.3 Detailed Package Install Instructions:

(1) Install the dependent packages

(2) Go to the directory of GMAT and type

```
> python setup.py install
```

3 REMMAX function

Rapid Epistatic Mixed Model Association Studies

Cite:

* Dan Wang, Hui Tang, Jian-Feng Liu, Shizhong Xu, Qin Zhang and Chao Ning. Rapid Epistatic Mixed Model Association Studies by Controlling Multiple Polygenic Effects. BioRxiv, 2020. doi: https://doi.org/10.1101/2020.03.05.976498

* Chao Ning, Dan Wang, Huimin Kang, Raphael Mrode, Lei Zhou, Shizhong Xu, Jian-Feng Liu. A rapid epistatic mixed-model association analysis by linear retransformations of genomic estimated values. Bioinformatics, 2018, 34(11): 1817-1825.

3.1 Format of the input file.

- Plink binary file including *.bed, *.bim and *.fam.
  Missing genotypes are recommended to impute with Beagle or other softwares, although they will be imputed according the frequency of occurrence locus by locus.

- phenotypic file:

(1) Delimited by blanks or tabs;

(2) All individuals in the plink file must have phenotypic values. If no, please remove these individuals from the plink binary file;

(3) The fisrt column is the family id and the second column is the individual id. The first two columns are the same to plink fam file, but order can be different;

(4) The last column is the phenotypic values. **Miss values are not allowed**;

(5) The covariates (including population means) are put before the phenotypic column. A column of 1's must be contained.
An example phenotypic file with four covariates (population mean, sex, age, treatmeant or untreatmeant) is as follows:

```
12659 14462 1 0 126 0 0.58
12659 14463 1 0 91 1 0.39
12659 14464 1 1 126 0 0.37
12659 14465 1 0 91 1 0.9
12659 14466 1 0 91 1 0.84
12659 14467 1 0 91 1 0.61
12659 14468 1 1 91 1 0.84
```

3.2 Exhaustive additive by addtive epistatis

Data: Mouse data in directory of GMAT/examples/data/mouse

3.2.1 Example 1: Include additive and additive by additive genomic relationship matrix

1. Exact test (for small data)

[ ]:

```
import logging
import numpy as np
import pandas as pd
from gmat.gmatrix import agmat, dgmat_as
from gmat.uvlmm.design_matrix import design_matrix_wemai_multi_gmat
from gmat.uvlmm.uvlmm_varcom import wemai_multi_gmat
from gmat.remma.remma_cpu.remma_epiAA_cpu import remma_epiAA_cpu,␣
 ↪remma_epiAA_cpu_parallel
from gmat.remma.annotation import annotation_snp_pos
logging.basicConfig(level=logging.INFO)

pheno_file = 'pheno'
bed_file = 'plink'

# Step 1: Calculate the genomic relationship matrix
agmat_lst = agmat(bed_file, inv=False) # additive genomic relationship matrix
dgmat_lst = dgmat_as(bed_file, inv=False) # dominace genomic relationship␣
 ↪matrix

# Step 2: Prepare the phenotypic vector (y), designed matrix for fixed effects␣
 ↪(xmat) and designed matrix for random effects (zmat)
y, xmat, zmat = design_matrix_wemai_multi_gmat(pheno_file, bed_file)

# Step 3: Estimate the variances
gmat_lst = [agmat_lst[0], agmat_lst[0]*agmat_lst[0]]  #␣
 ↪agmat_lst[0]*agmat_lst[0] is the additive by additive genomic relationship␣
 ↪matrix
var_com_a_axa = wemai_multi_gmat(y, xmat, zmat, gmat_lst)
print(var_com_a_axa)  # a list [0] addtive variance; [1] additive by additive␣
 ↪variance; [2] residual variance

# Step 4: Test
remma_epiAA_cpu(y, xmat, zmat, gmat_lst, var_com=var_com_a_axa,␣
 ↪bed_file=bed_file, p_cut=0.0001, out_file='remma_epiAA_cpu')

# Step 5: Select top SNPs and add the SNP position
res_file = 'remma_epiAA_cpu'  # result file
annotation_snp_pos(res_file, bed_file, p_cut=1.0e-5)
```

parallel Analysis can be subdivided with remma_epiAA_cpu_parallel and run parallelly on different machines.

[ ]:

```
# Step 1-3 is same to the above
```

```python
# Step 4: parallel test. Write the codes in separate scripts and run separately.
from gmat.remma.remma_cpu.remma_epiAA_cpu import remma_epiAA_cpu_parallel
remma_epiAA_cpu_parallel(y, xmat, zmat, gmat_lst, var_com=var_com_a_axa,␣
 ↪bed_file=bed_file, parallel=[3,1],
                         p_cut=0.0001, out_file='remma_epiAA_cpu')
remma_epiAA_cpu_parallel(y, xmat, zmat, gmat_lst, var_com=var_com_a_axa,␣
 ↪bed_file=bed_file, parallel=[3,2],
                         p_cut=0.0001, out_file='remma_epiAA_cpu')
remma_epiAA_cpu_parallel(y, xmat, zmat, gmat_lst, var_com=var_com_a_axa,␣
 ↪bed_file=bed_file, parallel=[3,3],
                         p_cut=0.0001, out_file='remma_epiAA_cpu')

# Step 5: Merge files 'remma_epiAA_cpu.1', 'remma_epiAA_cpu.2' and␣
 ↪'remma_epiAA_cpu.3' with the following codes.
prefix = 'remma_epiAA_cpu'
parallel_num = 3  # the number of parallels
with open(prefix + ".merge", 'w') as fout:
    with open(prefix + '.1') as fin:
        head_line = fin.readline()
        fout.write(head_line)
    for i in range(1, 4):
        with open(prefix + '.' + str(i)) as fin:
            head_line = fin.readline()
            for line in fin:
                fout.write(line)

# Step 6: Select top SNPs and add the SNP position
res_file = 'remma_epiAA_cpu.merge'  # result file
annotation_snp_pos(res_file, bed_file, p_cut=1.0e-5)
```

2. approximate test (recommended for big data)

[ ]:

```python
import logging
import numpy as np
import pandas as pd
from scipy.stats import chi2
from gmat.gmatrix import agmat, dgmat_as
from gmat.uvlmm.design_matrix import design_matrix_wemai_multi_gmat
from gmat.uvlmm.uvlmm_varcom import wemai_multi_gmat
from gmat.remma.random_pair import random_pair
from gmat.remma.remma_cpu.remma_epiAA_cpu import remma_epiAA_pair_cpu,␣
 ↪remma_epiAA_eff_cpu_c, remma_epiAA_eff_cpu_c_parallel
from gmat.remma.annotation import annotation_snp_pos
logging.basicConfig(level=logging.INFO)

pheno_file = 'pheno'
bed_file = 'plink'
```

```
# Step 1-3 is same to exact test

# Step 4: Randomly select 100,000 SNP pairs
snp_df = pd.read_csv(bed_file + '.bim', header=None, sep='\s+')
num_snp = snp_df.shape[0]  # the number of snp
random_pair(num_snp, out_file='random_pair', num_pair=100000,␣
 ↪num_each_pair=5000)

# step 5: Test these 100,000 SNP pairs
# note: set p_cut=1 to save all the results
remma_epiAA_pair_cpu(y, xmat, zmat, gmat_lst, var_com=var_com_a_axa,␣
 ↪bed_file=bed_file, snp_pair_file="random_pair",
                     max_test_pair=50000, p_cut=1,␣
 ↪out_file='remma_epiAA_pair_cpu_random')

# step 6: Calculate the median of variances for estimated epistatic SNP effects
res_df = pd.read_csv('remma_epiAA_pair_cpu_random', header=0, sep='\s+')
print(np.median(res_df['p']))  # P value close to 0.5. It means type I error␣
 ↪controlled well
var_median = np.median(res_df['var'])  # median of variances for estimated␣
 ↪epistatic SNP effects

# step 7: Screen the effects and select top SNP pairs based on approximate test.
 ↪
# Use the above median of variances as the approximate values (var_app =␣
 ↪var_median)
remma_epiAA_eff_cpu_c(y, xmat, zmat, gmat_lst, var_com=var_com_a_axa,␣
 ↪bed_file=bed_file, var_app=var_median,
                      p_cut=1e-05, out_file='remma_epiAA_eff_cpu_c')

# Step 8: Calculate exact p values for top SNP pairs
remma_epiAA_pair_cpu(y, xmat, zmat, gmat_lst, var_com=var_com_a_axa,␣
 ↪bed_file=bed_file, snp_pair_file="remma_epiAA_eff_cpu_c",
                     max_test_pair=50000, p_cut=1,␣
 ↪out_file='remma_epiAA_pair_cpu_res')

# Step 9: Select top SNPs and add the SNP position
res_file = 'remma_epiAA_pair_cpu_res'  # result file
annotation_snp_pos(res_file, bed_file, p_cut=1.0e-5)
```

parallel Analysis can be subdivided with remma_epiAA_eff_cpu_c_parallel and run parallelly on different machines.

[ ]:
```
# Step 1-6 is same to the above
# Step 7: parallel test. Write the codes in separate scripts and run separately.
from gmat.remma.remma_cpu.remma_epiAA_cpu import remma_epiAA_eff_cpu_c_parallel
remma_epiAA_eff_cpu_c_parallel(y, xmat, zmat, gmat_lst, var_com=var_com_a_axa,␣
 ↪bed_file=bed_file, parallel=[3,1],
                               var_app=var_median, p_cut=1.0e-5,␣
 ↪out_file='remma_epiAA_eff_cpu_c')
remma_epiAA_eff_cpu_c_parallel(y, xmat, zmat, gmat_lst, var_com=var_com_a_axa,␣
 ↪bed_file=bed_file, parallel=[3,2],
                               var_app=var_median, p_cut=1.0e-5,␣
 ↪out_file='remma_epiAA_eff_cpu_c')
remma_epiAA_eff_cpu_c_parallel(y, xmat, zmat, gmat_lst, var_com=var_com_a_axa,␣
 ↪bed_file=bed_file, parallel=[3,3],
                               var_app=var_median, p_cut=1.0e-5,␣
 ↪out_file='remma_epiAA_eff_cpu_c')

# Step 8: Calculate exact p values for top SNP pairs
remma_epiAA_pair_cpu(y, xmat, zmat, gmat_lst, var_com=var_com_a_axa,␣
 ↪bed_file=bed_file, snp_pair_file="remma_epiAA_eff_cpu_c.1",
                     max_test_pair=50000, p_cut=1,␣
 ↪out_file='remma_epiAA_pair_cpu_res.1')
remma_epiAA_pair_cpu(y, xmat, zmat, gmat_lst, var_com=var_com_a_axa,␣
 ↪bed_file=bed_file, snp_pair_file="remma_epiAA_eff_cpu_c.2",
                     max_test_pair=50000, p_cut=1,␣
 ↪out_file='remma_epiAA_pair_cpu_res.2')
remma_epiAA_pair_cpu(y, xmat, zmat, gmat_lst, var_com=var_com_a_axa,␣
 ↪bed_file=bed_file, snp_pair_file="remma_epiAA_eff_cpu_c.3",
                     max_test_pair=50000, p_cut=1,␣
 ↪out_file='remma_epiAA_pair_cpu_res.3')

# Step 9: Merge files 'remma_epiAA_pair_cpu_res.1', 'remma_epiAA_pair_cpu_res.
 ↪2' and 'remma_epiAA_pair_cpu_res.3'
# with the following codes.
prefix = 'remma_epiAA_pair_cpu_res'
parallel_num = 3  # the number of parallels
with open(prefix + ".merge", 'w') as fout:
    with open(prefix + '.1') as fin:
        head_line = fin.readline()
        fout.write(head_line)
    for i in range(1, 4):
        with open(prefix + '.' + str(i)) as fin:
            head_line = fin.readline()
            for line in fin:
                fout.write(line)

# Step 10: Select top SNPs and add the SNP position
res_file = 'remma_epiAA_pair_cpu_res.merge'  # result file
```

```
annotation_snp_pos(res_file, bed_file, p_cut=1.0e-5)
```