**Simulation code for Jiang et al (2020)**

Copyright 2020, Georgetown University and University of Pittsburgh. All Rights Reserved.

**The PedGFLMM R package**

If you would like to apply our statistics yourself, please use the PedGFLMM R package, which is available at <https://github.com/DanielEWeeks/PedGFLMM>

Note also that within the PedGFLMM R package, we provide an illustrative example of how to apply our statistical functions to real data as reformatted and imported via our Mega2 C program and Mega2R R package (https://cran.r-project.org/package=Mega2R). We suggest that would be an easier way for the interested reader to apply our statistics to their own data in an automated manner, across genes or user-defined regions.

**Simulation code**

Here we present example simulation code that was used to generate results in our paper:

Jiang YD, Chiu CY, Yan Q, Chen W, Gorin MB, Conley YP, Lakhal-Chaieb ML, Cook RJ, Amos CI, Wilson AF, Bailey-Wilson JE, McMahon FJ, Vazquez AI, Yuan A, Zhong XG, Xiong MM, Weeks DE, and Fan RZ (2020) Gene-based association testing of dichotomous traits with generalized linear mixed models for family data.

In case of suggestions and questions and/or problems, please contact us via e-mail ([rf740@georgetown.edu](mailto:rf740@georgetown.edu)).

For a mapping of the various functions to the results in the paper, please see Table 1 (below on page 11).

In this repository, we have this directory structure:

.

├── PedGFLMM

│   └── data

├── PedGFLMM\_non\_dynamic

│   └── data

├── analysis\_code

└── simulations

├── COSI\_50\_Ped

│   ├── Type\_I\_Error\_Rare

│   ├── Type\_I\_Error\_Rare\_and\_Common

│   ├── power\_codes

│   │   └── test\_C=1\_Delta=0.5

│   └── power\_test\_C=1\_Delta=0.50

│   ├── Fig\_Pow

│   ├── Fig\_Pow\_orig

│   ├── Power\_causal\_rare

│   ├── Power\_causal\_rare\_and\_common

├── Input

├── SKAT

│   └── Simulation\_Code\_Example

└── Type\_I\_error\_rate\_aggregate

There are four main sub-directories:

**1. The ‘PedGFLMM’ directory**

PedGFLMM

├── MGAO.R

├── PedGFLMM\_EXAMPLE.R

├── PedGFLMM\_beta\_smooth\_only.R

├── PedGFLMM\_doc.pdf

├── PedGFLMM\_fixed\_model.R

├── PedGLMM\_additive\_effect\_model.R

├── data

│   ├── Ped.csv

│   ├── covariate.csv

│   ├── geno.csv

│   └── snpPos.csv

└── dynamic\_rule.R

The ‘PedGFLMM’ directory contains the version of the PedGFLMM codes that were used in the simulations. These were reformatted and improved to form our well-documented ‘PedGFLMM’ R package, which is available at <https://github.com/DanielEWeeks/PedGFLMM>. The less-well-documented version of our code in the ‘PedGFLMM’ directory is provided here because these are the codes that were used by our simulation codes provided in the ‘simulations’ directory.

We encourage users to use the polished well-documented ‘PedGFLMM’ R package instead of the rougher code presented here.

In addition to providing the codes, in the 'data' subfolder, we provide an example toy data set in the format expected by our functions. The PedGFLMM R vignette clearly describes the expected format of these input data.

**2. The ‘simulations’ directory**

The ‘simulations’ directory contains the simulation codes, as well as some results of the simulations, along with the code used to generate the corresponding Figures and Table.

Directory structure of the ‘simulations’ folder:

simulations

├── COSI\_50\_Ped

│   ├── Type\_I\_Error\_Rare

│   ├── Type\_I\_Error\_Rare\_and\_Common

│   ├── power\_codes

│   │   └── test\_C=1\_Delta=0.5

│   └── power\_test\_C=1\_Delta=0.50

│   ├── Fig\_Pow

│   ├── Fig\_Pow\_orig

│   ├── Power\_causal\_rare

│   ├── Power\_causal\_rare\_and\_common

├── Input

├── SKAT

│   └── Simulation\_Code\_Example

└── Type\_I\_error\_rate\_aggregate

**3. The 'PedGFLMM\_non\_dynamic' directory**

PedGFLMM\_non\_dynamic/

├── PedGFLMM\_EXAMPLE.R

├── PedGFLMM\_beta\_smooth\_only.R

├── PedGFLMM\_doc.pdf

├── PedGFLMM\_fixed\_model.R

├── PedGLMM\_additive\_effect\_model.R

└── data

├── Ped.csv

├── covariate.csv

├── geno.csv

└── snpPos.csv

The 'PedGFLMM\_non\_dynamic' directory contains functions that were used in the simulations where we did not use the dynamic rule to set the number of basis functions. These functions can easily be used in the simulation framework below by calling these files instead of the ones in the "PedGFLMM" directory. For more details about these specific functions, see the "PedGFLMM\_doc.pdf" file in this directory.

**4. The ‘analysis\_code’ directory**

analysis\_code/

├── README.md

├── data\_analysis.R

└── dynamic\_rule.R

The ‘analysis\_code’ directory contains a copy of the code we used to analyze the real data. Note that a modified dynamic rule, as defined by the `dynamic\_rule.R` file in this folder, was used to analyze the real data.

While our complete real data set is unavailable due to consent issues, we have included our real analysis code in this GitHub repository as a model. However, actually within the PedGFLMM R package, we provide an illustrative example of how to apply our statistical functions to real data as reformatted and imported via our Mega2 and Mega2R R package (https://cran.r-project.org/package=Mega2R), and would suggest that would be an easier way for the interested reader to apply our statistics to their own data. Please see the PedGFLMM R package vignette for further details.

**Type I Error and Power Simulations**

The code used for the Type I Error and Power Simulations is documented below. For additional details of how the simulation studies were done, please see the "Simulation Studies" section of our Jiang et al. paper (citation above).

For the calculations described below to get power levels and type I error rates, one has to run the R codes on a Unix/Linux machine (or on a Macintosh). Our simulations were originally carried out on the helix/biowulf computational cluster at NIH.

***Type I Error Simulations Timing:***

For the Type I Error study, to complete all simulations to get the Table 2, it would have taken about 5 days if we were able to start all jobs in parallel on our computer cluster at the same time. Note, we did 400 seeds each with 2,500 replicates to have a total of 3 million replicates for each case. The shortest time was 50 hours for the case of 6 kb Rare case, and the longest was 104 hours for the case of 21 kb Rare and common case. In practice, it took us almost 2 weeks to get all results for Table 2, since we were not able to start all jobs at the same time.

On an Intel Xeon E5-2690 v3 2.6 GHz processor, using the code in the "Type\_I\_error\_Rare\_and\_Common\_region\_size\_6k.R", each replicate took approximately 101.8 seconds to complete. Similarly based on running "Type\_I\_error\_Rare\_region\_size\_6k.R" for 5 replicates, each replicate took approximately 111.6 seconds to complete.

**INSTALLATION**

**Compilation of C functions in the Util.c file**

├── simulations

│   ├── Make.sh

│   ├── Util.c

│   ├── Util\_Read\_Me.docx

Our simulation scripts rely on some C functions that are defined in the Util.c file, which is in the ‘simulations’ folder. Before the simulation scripts below can be run, the Util.c file has to be compiled to generate a ‘Util.so’ file. This works on Unix and Macintosh OS X machines.

On a Unix machine or in a Terminal window on a Macintosh OS X machine (with a C compiler installed), the ‘Util.so’ can be generated via these commands, to be issued in the 'simulations' folder:

chmod +x Make.sh

./Make.sh

**R libraries required for the simulations**

The simulation scripts can only be run after the following required R packages have been installed:

fda

MASS

Matrix

nlme

glmm

pedgene

pedigreemm

tidyverse

**Decompress haplotype file**

├── simulations

│   ├── SKAT

│   │   └── Simulation\_Code\_Example

│   │   ├── **out.50.hap-1.zip**

│   │   └── out.50.pos-1

To carry out the simulations below, first you must unzip the

"out.50.hap-1.zip"

file in the "simulations/SKAT/Simulation\_Code\_Example" folder.

**Reproducing the simulation studies**

We carried out simulations under a number of conditions and variations. Here we describe here how to reproduce the Type I and Power simulations using the dynamic rule as applied to the 50 pedigree template.

**Reproducing the Type I Error simulations**

├── simulations

│   ├── COSI\_50\_Ped

│   │   ├── Type\_I\_error\_Rare\_and\_Common\_region\_size\_6k.R

│   │   ├── Type\_I\_error\_Rare\_region\_size\_6k.R

│   │   ├── Type\_I\_error\_func.R

Type I error calculation: we provide two R scripts, Type\_I\_error\_Rare\_region\_size\_6k.R and Type\_I\_error\_Rare\_and\_Common\_region\_size\_6k.R, in the **simulations/COSI\_50\_Ped/** folder; these call the function in the Type\_I\_error\_func.R file. The two scripts should generate two files, one is Type\_I\_error\_region\_size\_6000\_Nsim\_2500\_seed\_1.csv in **simulations/COSI\_50\_Ped/Type\_I\_Error\_Rare**, and the other is Type\_I\_error\_region\_size\_6000\_Nsim\_2500\_seed\_1.csv in **simulations/COSI\_50\_Ped/Type\_I\_Error\_Rare\_and\_Common.**

For more results, one can change the seed number to generate them.

On an Intel Xeon E5-2690 v3 2.6 GHz processor, running Type\_I\_error\_Rare\_region\_size\_6k.R to simulate 501 replicates took 9.9 hours, about 71.2 seconds per replicate (prior to our initial submission; code should be faster now).

How to run *'Type\_I\_error\_Rare\_region\_size\_6k.*R'

This file is in the "simulations/COSI\_50\_Ped" folder.

Open it in RStudio and click on the ‘Source’ button.

Or start up R and source the file via this R command:

source("Type\_I\_error\_Rare\_region\_size\_6k.R")

The 'Type\_I\_error\_Rare\_and\_Common\_region\_size\_6k.R' R script can be run in the same manner.

Output of the Type I Error simulation programs:

The 'Type\_I\_error\_Rare\_region\_size\_6k.R' R script writes its output into the 'simulations/COSI\_50\_Ped/Type\_I\_Error\_Rare' folder.

The 'Type\_I\_error\_Rare\_and\_Common\_region\_size\_6k.R' R script writes its output into the 'simulations/COSI\_50\_Ped/Type\_I\_Error\_Rare\_and\_Common**'** folder.

**Reproducing the Power simulations**

Power level calculation: we provide twelve R scripts in **simulations/COSI\_50\_Ped/power\_codes/test\_C=1\_Delta=0.5**, which call the functionPower\_func.R. Then one should be able to generate all the power levels in **simulations/COSI\_50\_Ped/powe\_test\_C=1\_Delta=0.5**

power\_codes

   └── test\_C=1\_Delta=0.5

   ├── Power\_GFLMM\_rare\_and\_common\_region\_size\_15k\_18k\_21k\_Neg\_beta\_pct\_00.R

   ├── Power\_GFLMM\_rare\_and\_common\_region\_size\_15k\_18k\_21k\_Neg\_beta\_pct\_20.R

   ├── Power\_GFLMM\_rare\_and\_common\_region\_size\_15k\_18k\_21k\_Neg\_beta\_pct\_50.R

   ├── Power\_GFLMM\_rare\_and\_common\_region\_size\_6k\_9k\_12k\_Neg\_beta\_pct\_00.R

   ├── Power\_GFLMM\_rare\_and\_common\_region\_size\_6k\_9k\_12k\_Neg\_beta\_pct\_20.R

   ├── Power\_GFLMM\_rare\_and\_common\_region\_size\_6k\_9k\_12k\_Neg\_beta\_pct\_50.R

   ├── Power\_GFLMM\_rare\_region\_size\_15k\_18k\_21k\_Neg\_beta\_pct\_00.R

   ├── Power\_GFLMM\_rare\_region\_size\_15k\_18k\_21k\_Neg\_beta\_pct\_20.R

   ├── Power\_GFLMM\_rare\_region\_size\_15k\_18k\_21k\_Neg\_beta\_pct\_50.R

   ├── Power\_GFLMM\_rare\_region\_size\_6k\_9k\_12k\_Neg\_beta\_pct\_00.R

   ├── Power\_GFLMM\_rare\_region\_size\_6k\_9k\_12k\_Neg\_beta\_pct\_20.R

   └── Power\_GFLMM\_rare\_region\_size\_6k\_9k\_12k\_Neg\_beta\_pct\_50.R

*Power Simulations Timing:*

On an Intel Xeon E5-2690 v3 2.6 GHz processor, running only the first call to the power function "power\_func" in the Power\_GFLMM\_rare\_and\_common\_region\_size\_15k\_18k\_21k\_Neg\_beta\_pct\_00.R file to simulate 100 replicates took 2.59 hours, about 93.2 seconds per replicate.

For the power comparisons with the revised code, we set the upper limit of the total number of replicates as 3,000 and stop the simulation if all of our proposed methods get 1,000 valid samples (without any error or warnings). Note, in "power\_codes", each job files includes nine cases of simulations (e.g., calls to the "power\_func" function). When we did the computer experiments, we split each job files into three, which includes three simulations only. This took us almost 5 days to get all results.

How to run the 'Power\_GFLMM\_rare\_and\_common\_region\_size\_15k\_18k\_21k\_Neg\_beta\_pct\_00.R' power simulation script:

Open it in RStudio and click on the ‘Source’ button.

Or start up R and source the file via this R command:

source("Power\_GFLMM\_rare\_and\_common\_region\_size\_15k\_18k\_21k\_Neg\_beta\_pct\_00.R")

The other R scripts can be run in the same manner.

Output of the Power simulation programs:

Each of the power simulation R scripts writes its output into one of the two subfolders

Power\_causal\_rare

Power\_causal\_rare\_and\_common

in the **simulations/COSI\_50\_Ped/powe\_test\_C=1\_Delta=0.5** folder.

**Reproducing the Figures**

Here we provide and document the code that we used to generate the bar plot figures based on the 50 pedigree power simulations.

├── simulations

│   ├── bar\_dynamic\_function\_plot.R

│   ├── bar\_dynamic\_result\_plot.R

Using as input the results of our previous (time-consuming) simulations, one may source *bar\_dynamic\_result\_plot.R* in the **simulations** folder to generate Figures in the **simulations/COSI\_50\_Ped/power\_test\_C=1\_Delta=0.50/Fig\_Pow/** folder, which are the Figures 1 – 12 in the main text. If you have not carried out any simulations, then these should match the original set of Figures which are in the **simulations/COSI\_50\_Ped/power\_test\_C=1\_Delta=0.50/Fig\_Pow\_orig** folder. If you carry out simulations above, these may change some of the results in the simulation output directories which form the input data for these Figures – in such a case, if you desire to use our original input, just download everything again.

How to run ‘*bar\_dynamic\_result\_plot.R'*:

Open it in RStudio and click on the ‘Source’ button.

Or start up R and source the file via this R command:

source("bar\_dynamic\_result\_plot.R ")

Input for the ‘*bar\_dynamic\_result\_plot.R’* Figure-generating program:

Within the ‘simulations/COSI\_50\_Ped/power\_test\_C=1\_Delta=0.50’ folder, there are two folders ‘Power\_causal\_rare’ and ‘Power\_causal\_rare\_and\_common’ which contain the results of our previous (time-consuming) simulations.

Output of the ‘*bar\_dynamic\_result\_plot.R’* Figure-generating program:

When this R program is run by being sourced, the resulting Figures are output to the ‘simulations/COSI\_50\_Ped/power\_test\_C=1\_Delta=0.50/Fig\_Pow’ folder. These should match the original set of Figures which are in the ‘simulations/COSI\_50\_Ped/power\_test\_C=1\_Delta=0.50/Fig\_Pow\_orig’ folder.

**Reproducing Table 2 - the empirical type I error rates**

To reproduce Table 2, you first need to run the Type\_I\_error\_Rare\_region\_size\_6k.R and Type\_I\_error\_Rare\_and\_Common\_region\_size\_6k.R scripts described above to properly populate the **simulations/COSI\_50\_Ped/Type\_I\_Error\_Rare** and **simulations/COSI\_50\_Ped/Type\_I\_Error\_Rare\_and\_Common** folders with the desired set of results.

Note that the set of results you obtain may differ a bit from those presented in the paper because these are simulations that depend not only on the exact set of random number seeds used but may also depend on the machine on which it is run.

Then, to calculate your empirical type I error rates in Table 2 for 6 kb region based on your results of those (time-consuming) simulations, source the *50\_Ped\_result\_Type\_I\_aggregate.R* file in **simulations/Type\_I\_error\_rate\_aggregate,** and the results will be generated in the **simulations/COSI\_50\_Ped/Type\_I\_Error\_combined\_Results** folder.

How to run ‘*50\_Ped\_result\_Type\_I\_aggregate.R’*:

After you have properly populated the the **simulations/COSI\_50\_Ped/Type\_I\_Error\_Rare** and **simulations/COSI\_50\_Ped/Type\_I\_Error\_Rare\_and\_Common** folders with the desired set of results, then you can proceed as follows:

Open it in RStudio and click on the ‘Source’ button.

Or start up R and source the file via this R command:

source("50\_Ped\_result\_Type\_I\_aggregate.R’")

Note: When this file is sourced, it will generate these messages, which can be disregarded:

ERROR : cannot open the connection

Warning message:

In file(file, "rt") :

cannot open file 'COSI\_50\_Ped/Type\_I\_Error\_Rare\_and\_Common/Type\_I\_error\_region\_size\_6000\_Nsim\_2500\_seed\_48.csv': No such file or directory

Input for the ‘50\_Ped\_result\_Type\_I\_aggregate.R’ function:

The input files, containing the results of our previous simulations, are located in the two folders ‘Type\_I\_Error\_Rare’ and ‘Type\_I\_Error\_Rare\_and\_Common’ within the ‘simulations/COSI\_50\_Ped’ folder.

Output of the ‘50\_Ped\_result\_Type\_I\_aggregate.R’ function:

The output will be generated in the two sub-folders ‘Type\_I\_Error\_Rare’ and ‘Type\_I\_Error\_Rare\_and\_Common’ of the ‘simulations/COSI\_50\_Ped/Type\_I\_Error\_combined\_Results’ folder.

**Table 1:** Mapping of the functions to the results in the paper

|  |  |  |
| --- | --- | --- |
| **Script file name** | **Function** | **How it relates to the manuscript** |
| MGAO.R | Perform the principal component analysis discussed by Gao et al., 2008 | Evaluate the dimension of genotype data to determine the number of basis functions |
| PedGFLMM\_EXAMPLE.R | Provide working examples to build the GFLMM | Give illustrative examples |
| PedGFLMM\_beta\_smooth\_only.R | Estimate the GFLMM statistics under the beta-smooth only model | Implement Equation (2), revised by Equation (6) |
| PedGFLMM\_fixed\_model.R | Estimate the GFLMM statistics under the fixed model | Implement Equation (1), revised by Equation (5) |
| PedGLMM\_additive\_effect\_model | Estimate the GFLMM statistics under the additive model | Implement Equation (3) |
| dynamic\_rule.R | Apply the dynamic rule to determine the functional data analysis parameters | Implement the method discussed in Functional Data Analysis Parameters and Dynamic Rule |
| Type\_I\_error\_Rare\_and\_Common\_region\_size\_6k.R | Simulate type I error with a mix of rare and common genetic variants within a 6 kb region | Generate type I error rates in Table 2 |
| Type\_I\_error\_Rare\_region\_size\_6k.R | Simulate type I error with rare genetic variants within a 6 kb region | Generate type I error rates in Table 2 |
| Type\_I\_error\_func.R |  |  |
| Power\_GFLMM\_rare\_and\_common\_region\_size\_6k\_9k\_12k\_Neg\_beta\_pct\_00.R | Simulate power with a mix of rare and common genetic variants with 0% negative-effect causal variants within a 6 kb, 9 kb, or 12 kb region | Generate empirical power in Figure 1-3 |
| Power\_GFLMM\_rare\_and\_common\_region\_size\_6k\_9k\_12k\_Neg\_beta\_pct\_20.R | Simulate power with a mix of rare and common genetic variants with 20% negative-effect causal variants within a 6 kb, 9 kb, or 12 kb region | Generate empirical power in Figure 1-3 |
| Power\_GFLMM\_rare\_and\_common\_region\_size\_6k\_9k\_12k\_Neg\_beta\_pct\_50.R | Simulate power with a mix of rare and common genetic variants with 50% negative-effect causal variants within a 6 kb, 9 kb, or 12 kb region | Generate empirical power in Figure 1-3 |
| Power\_GFLMM\_rare\_and\_common\_region\_size\_15k\_18k\_21k\_Neg\_beta\_pct\_00.R | Simulate power with a mix of rare and common genetic variants with 0% negative-effect causal variants within a 15 kb, 18 kb, or 21 kb region | Generate empirical power in Figure 4-6 |
| Power\_GFLMM\_rare\_and\_common\_region\_size\_15k\_18k\_21k\_Neg\_beta\_pct\_20.R | Simulate power with a mix of rare and common genetic variants with 20% negative-effect causal variants within a 15 kb, 18 kb, or 21 kb region | Generate empirical power in Figure 4-6 |
| Power\_GFLMM\_rare\_and\_common\_region\_size\_15k\_18k\_21k\_Neg\_beta\_pct\_50.R | Simulate power with a mix of rare and common genetic variants with 50% negative-effect causal variants within a 15 kb, 18 kb, or 21 kb region | Generate empirical power in Figure 4-6 |
| Power\_GFLMM\_rare\_region\_size\_6k\_9k\_12k\_Neg\_beta\_pct\_00.R | Simulate power with rare genetic variants with 0% negative-effect causal variants within a 6 kb, 9 kb, or 12 kb region | Generate empirical power in Figure 7-9 |
| Power\_GFLMM\_rare\_region\_size\_6k\_9k\_12k\_Neg\_beta\_pct\_20.R | Simulate power with rare genetic variants with 20% negative-effect causal variants within a 6 kb, 9 kb, or 12 kb region | Generate empirical power in Figure 7-9 |
| Power\_GFLMM\_rare\_region\_size\_6k\_9k\_12k\_Neg\_beta\_pct\_50.R | Simulate power with rare genetic variants with 50% negative-effect causal variants within a 6 kb, 9 kb, or 12 kb region | Generate empirical power in Figure 7-9 |
| Power\_GFLMM\_rare\_region\_size\_15k\_18k\_21k\_Neg\_beta\_pct\_00.R | Simulate power with rare genetic variants with 0% negative-effect causal variants within a 15 kb, 18 kb, or 21 kb region | Generate empirical power in Figure 10-12 |
| Power\_GFLMM\_rare\_region\_size\_15k\_18k\_21k\_Neg\_beta\_pct\_20.R | Simulate power with rare genetic variants with 20% negative-effect causal variants within a 15 kb, 18 kb, or 21 kb region | Generate empirical power in Figure 10-12 |
| Power\_GFLMM\_rare\_region\_size\_15k\_18k\_21k\_Neg\_beta\_pct\_50.R | Simulate power with rare genetic variants with 50% negative-effect causal variants within a 15 kb, 18 kb, or 21 kb region | Generate empirical power in Figure 10-12 |
| bar\_dynamic\_function\_plot.R | Create the bar charts with 5%, 10%, and 15% causal genetic variants | Plot the bar charts illustrated by Figure 1-12 |
| bar\_dynamic\_result\_plot.R | Provide existing results to generate the bar charts | Provide data source as an illustrative example to create the bar charts illustrated by Figure 1-12 |
| data\_analysis.R | Apply the GFLMM to the real age-related macular degeneration data | Generate the test statistics in Table 3 |

Copyright 2020, Georgetown University and University of Pittsburgh. All Rights Reserved.