README:

(Updated January 27, 2016)

This is a readme file to explain this repository.

This covers the basic of using PrivSTRAT, PrivLMM, how to generate the figures from our project.

Idea of project:

This project aims at implementing tools to allow differentially private GWAS statistics while correcting for population stratification in the samples. In order to do this we have generated two methods, PrivLMM based on LMM, and PrivSTRAT based on EIGENSTRAT. Details of both methods to appear!

Note that PrivSTRAT is more thoroughly tested than PrivLMM, so users of PrivLMM be warned!

Running PrivSTRAT:

PrivSTRAT can perform three different tasks: picking high scoring SNPs, estimating the number of high scoring SNPs, and estimating the Wald test value of a given SNP.

The flag –t is used to signify which task. You can either use:
–t Count
-t Top
-t Wald
if not specified uses Top

The flag –e is used to specify the privacy budget (aka epsilon). For a privacy budget of 2.0 add
–e 2.0
If not specified uses epsilon=1.0

There is also the –k flag that tells the algorithm how many PCs to use for correction. The default is 5.

The –bed flag is given before the name of the binary ped file containing the genotype and phenotype information. For example, if group.bed/group.fam/group.bim are the files, write:

-bed group

The –save tag is used to specify a file to save to. For example, to save to savefile.txt write:

-save savefile.txt

If not specified nothing is saved.

The other flags are all task specific.

Picking high scoring SNPs:

For this task we need to specify the number of SNPs to return. This is done by the -mret flag. For example, if one wants the top 10 SNP use

-mret 10

Defaults to mret=3.

One can also specify the algorithm--either the neighbor distance based, noise (aka Laplacian) based or score based (see manuscript)—using the –a tag. Can use either
-a score
-a noise
-a neighbor

Default is set to neighbor. We do not recommend changing this setting, we include the score and noise based methods only for completeness.

Example: Assume we want to run on group.bed, with privacy budget epsilon=1.0, returning the top 10 SNPs, with 10 PCs, and the noise algorithm. Then we would run:

python PrivSTRAT.py –t Top -a noise –bed group –k 10 –mret 10 –e 1.0

Estimating number of significant SNPs:

We want to estimate the number of SNPs with chi^2 val > some threshold, where the chi 2 threshold is set using the –p tag. For example, for threshold 10.0 use:

-p 10.0

Example: Assume we want to run on group.bed, with privacy budget epsilon=1.0, estimating the number of SNPs with chi^2 val > 10.0, with 10 PCs. Then we would run:

python PrivSTRAT.py –t Count –bed group –k 10 –p 10.0 –e 1.0

Defaults to .05. Will update to take p values instead of chi^2 values soon.

Estimating Wald:

Finally, consider the tasks of estimating the Wald statistic. To do this we need to specify the SNPs, using –s tag. For example, if we want it for SNPs snp1 and snp2 write:

-s snp1 snp2

Example: Assume we want to run on group.bed, with privacy budget epsilon=1.0, estimating the Wald statistic on rs101 and rs102, with 10 PCs. Then we would run:

python PrivSTRAT.py –t Wald –bed group –k 10 –s rs101 rs102 –e 1.0

Divides the privacy budget evenly between all SNPs in the list.

Running PrivLMM:

PrivLMM.py is almost identical to PrivSTRAT.py, except for two main differences. First, instead of specifying as –k flag you specify –se2 and –sg2, the variance components.

Example: Assume we want to run on group.bed, with privacy budget epsilon=1.0, estimating the LLM based Wald statistic on rs101 and rs102, with sigma_e^2=.5 and sigma_g^2=.5. Then we would run:

python PrivLMM.py –t Wald –bed group –se2 .5 –sg2 .5 –s rs101 rs102 –e 1.0
In addition to that, the –t flag has one more option: Herit This returns estimates of sigma_e^2 and sigma_g^2

Estimating Heritability:

To estimate heritability using PrivLMM, we need to specify a –num parameter (the number of subsets used to estimate). If not specified is set equal to 5.

Example: Assume we want to run on group.bed, with privacy budget epsilon=1.0, estimating the heritability with num set to 10. Then we would run:

python PrivLMM.py –t Herit –bed group –num 10 –e 1.0

Generating Figures:

Note that our paper has numerous figures. The code in Top_STRAT_Fig.py and Top_LMM_Fig.py can be used to produce figures comparing the accuracy of the three algorithms (neighbor, noise and score) for picking top SNPs using PrivSTRAT and PrivLMM. WaldFig.py does something similar for the Wald test.

More details:

First consider the task of picking high scoring SNPs. Consider Top_STRAT_Fig.py, the way that Top_LMM_Fig.py works being similar.

At the bottom of the file Top_STRAT_Fig.py, right under the

if __name__==""__main__":

line, we see a few arguments that are hard coded in. The first is filename, which is the name of the Bed file being used. There are then two lists, eps and mret, of equal length. The idea is that, for each entry in mret, this code will compare the performance of the noise, neighbor, and score methods for picking high scoring SNPs. For mret[i], is compares these methods on different choices of epsilon ranging between eps[i] and 10*eps[i]. The results are then saved to a file, where the results of mret[i] are saved with a name whose prefix is given by savename followed by the suffix mret[i].txt.

This saved file contains 4 lines, the first listing the values in eps, and each other one being a list of the accuracy of a given algorithm (noise, neighbor or score) for the corresponding privacy parameter. This can then be plotted using matplotlib.pyplot

The WaldFig.py file has a similar setup, and plots the result as one line for each epsilon, with the other entries corresponding to the error at the 25^{th}, 50^{th}, and 75^{th} percentile respectively.

Overview of Important files/ classes:

MU_Mat.py: contains the class MU_Mat which is an interface that allows other files use to interact with the MU matrix in our method. Also contains the MU_Mem class, which is an extension of MU_Mat that assumes MU can be held in memory as a numpy array.

MU_LMM.py: Contains MU_LMM, a class that extends MU_Mem for PrivLMM.

MU_STRAT.py: Contains MU_STRAT, as class that extends MU_Mem for PrivSTRAT.

loadFile.py: Loads the supplied Bed file.

DP_util.py: The workhorse. Takes a MU_Mem object and uses it to perform the differentially private queries in our paper.

README.docx and README.pdf: Self explanatory.

PrivSTRAT.py: User interface for using PrivSTRAT method.

PrivLMM.py: User interface for using PrivSTRAT method.

PrivGWAS.py: Deals with some of the work that PrivLMM.py and PrivSTRAT.py have in common.

test.py: A few of the ways we tested the code—not all of them!

Top_STRAT_Fig.py, Top_LMM_Fig.py, WaldFig.py: See above.

UI.py and TopSNP.py: depreciated, early versions of some things.

caseStudy.py and testCase.py: caseStudy.py was used to perform a case study for the paper, while testCase.py serves as a helper function to caseStudy.py.