StocSum: stochastic summary statistics for whole genome sequencing studies

Han Chen and Nannan Wang

May 1, 2023

The StocSum package accompanies the paper "StocSum: stochastic summary statistics for whole genome sequencing studies". StocSum is a novel reference-panel-free statistical framework for generating, managing, and analyzing stochastic summary statistics using random vectors. It is implemented to various downstream applications, including single-variant tests, conditional association tests, gene-environment interaction tests, variant set tests, as well as meta-analysis and LD score regression tools.

1. Input

StocSum requires an object from fitting the null model using the glmm.kin function from the GMMAT package, and a genotype file in a GDS format. For variant set test, a user-defined marker group file is required. Specified formats of these files are described as follows.

1.1 Object

StocSum can perform various applications including single-variant tests, conditional association tests, gene-environment interaction tests, variant set tests, as well as meta-analysis and LD score regression tools. To fit the null model, the phenotype and covariates (include the environmental factors of interest) should be saved in a data frame. If the samples are related, the relatedness should be known positive semidefinite matrices Vk as an R matrix (in the case of a single matrix) or an R list (in the case of multiple matrices). Refer to the GMMAT user manual (https://cran.r-project.org/web/packages/GMMAT/vignettes/GMMAT.pdf) to learn the method of fitting the null model. The class of the object should be "glmmkin".

1.2 Genotypes

StocSum can take genotype files in the GDS format. Genotypes in Variant Call Format (VCF) and PLINK binary PED format can be converted to the GDS format using seqVCF2GDS and seqBED2GDS functions from the SeqArray package:

```
library(SeqArray)
SeqArray::seqVCF2GDS("VCF_file_name","GDS_file_name")
SeqArray::seqBED2GDS("BED_file_name","FAM_file_name","BIM_file_name","GDS_file_name")
```

1.3 Group definition file

For variant set test, a group definition file with no header and 6 columns (variant set id, variant chromosome, variant position, variant reference allele, variant alternate allele, weight) is required. For example, here we show the first 5 rows of the example group definition file "SetID.withweights.txt":

```
library(StocSum)
group.file <- system.file("extdata", "SetID.withweights.txt", package = "StocSum")
groups <- read.table(group.file)
head(groups)
```

```
##     V1 V2 V3 V4 V5 V6
## 1 Set1  1  1  T  A  1
## 2 Set1  1  2  A  C  4
## 3 Set1  1  3  C  A  3
## 4 Set1  1  4  G  A  6
## 5 Set1  1  5  A  G  9
## 6 Set1  1  6  C  A  9
```

Note that each variant in the group definition file is matched by chromosome, position, reference allele and alternate allele with variants from the GDS file. One genetic variant can be included in different groups with possibly different weights. If no external weights are needed in the analysis, simply replace the 6th column by all 1's.

2. Running StocSum

If StocSum has been successfully installed, you can load it in an R session using

```
library(StocSum)
```

We provide three functions in StocSum: StocSum.R to perform single-variant tests, conditional association tests, variant set tests, as well as meta-analysis; StocSum.GE.R to perform gene-environment interaction tests; StocSum.LDSC.R to perform LD score regression.

In StocSum.R, the nested function glmmin2randomvec is for generating the random vectors, the nested function G.stat is for calculating the stochastic summary statistics, the nested function svt.pval is for running single variant score tests, the nested function svt.meta is perform meta-analysis on score test results, the next functions G.prep and G.pval is two-step low-memory of performing variant set tests.

In StocSum.GE.R, the nested function glmmin2randomvec is for generating the random vectors from multivariate normal distribution with mean 0 and covariance matrix P; the nested function GE.stat is for calculating the stochastic summary statistics; the nested function GE.svt.pval is for running single variant score tests,

In StocSum.LDSC.R, the nested function LDSC.glmmin2randomvec is for generating the random vectors from multivariate normal distrubution with mean 0 and covariance matrix P=I-1(1^' 1)^(-1) 1'; the nested function G.stat is for calculating the stochastic summary statistics; the nested function LDSC.win is for calculating the LD scores.

Details about how to use these functions, their arguments and returned values can be found in the R help document of StocSum. For example, to learn more about glmmin2randomvec in an R session you can type

```
?glmmin2randomvec
```

2.1 Fitting GLMM

StocSum requires a "glmmkin" class object that contains a fitted GLMM null model. The object can be obtained from the glmmkin function from the R package GMMAT. For more examples and details about the glmmkin function, see the GMMAT manual (https://cran.r-project.org/web/packages/GMMAT/vignettes/GMMAT.pdf). Below is an example of fitting a GLMM using the glmmkin function from GMMAT:

```
library(StocSum)
library(GMMAT)
data(example)
attach(example)
GRM.file <- system.file("extdata", "GRM.txt.bz2", package = "StocSum")
GRM <- as.matrix(read.table(GRM.file, check.names = FALSE))
nullmod <- glmmkin(disease ~ age + sex, data = pheno, kins = GRM, id = "id", family = binomial(link = "
```

2.2 Single-variant tests

2.2.1 Generate random vectors To run StocSum.R, the user needs to generate the random vectors that having covariance approximate to the projection matrix from above fitted null model.

```
obj <- glmmkin2randomvec(nullmod)
```

The function glmmkin2randomvec returns a list. The element random.vectors stores the generated random vectors. The remaining elements theta, scaled.residuals, and id_include are inherited from the null model generated in 5.1.

If the returned nullmod object in 5.1 does no include the projection matrix P, the user needs to set the relation matrix in glmkin2randomvec function. The following is an example showing the setting when only the kinship is considered as the relation matrix.

```
kinship.chol <- chol(GRM)
obj <- glmmkin2randomvec(nullmod, Z = list(t(kinship.chol)))
```

2.2.2 Calculate summary statistics The next step is to calculate the summary statistics and stochastic summary statistics. The genotype file in the GDS format is as input. The output of G.stat is intermediates files containing single variant scores and the stochastic summary statistics. An example is as following:

```
out.prefix <- "test"
gdsfile <- system.file("extdata", "geno.gds", package = "StocSum")
G.stat(obj, geno.file = gdsfile, meta.file.prefix = out.prefix, MAF.range=c(0,0.5), miss.cutoff = 1)
```

```
## # of selected samples: 400
## # of selected variants: 100
```

The first argument "obj" in G.stat is the object returned by glmmkin2randomvec in section 5.2.2. The argument "geno.file" is the genotype file. The argument "meta.file.prefix" specifies the prefix of output files. In the example above, a space-delimited file "test.sample.1" will be generated to save the single variant scores, and a binary file "test.ressample.1" will be generated to save the stochastic summary statistics. Note that this binary file is not human-readable, but can be loaded by downstream modules/functions. The argument "ncores" in G.stat is to specify how many cores you would like to use on a computing node. It also determines the intermediate files. For example, if "ncores=2", there will be four intermediate files, i.e., "test.sample.1", "test.sample.2", "test.resample.1", "test.resample.2".

2.2.3 Calculating P-values When the intermediate files are generated by "G.stat", the function "svt.pval" can be used to calculate P-values for each variants. An example is as following:

```
out1<-svt.pval(out.prefix, MAF.range=c(0,0.5), miss.cutoff = 1, auto.flip=F)
```

The first argument "out.prefix" is the prefix of output files specified in "G.stat".

```

head(out1)
```

2.2.4 Output Files

```
##    SNP chr pos ref alt   N missrate   altfreq      SCORE       VAR      PVAL
## 1 SNP1   1   1   T   A 393   0.0175 0.9745547 -1.9849977  4.475883 0.3481136
## 2 SNP2   1   2   A   C 400   0.0000 0.5000000  3.5103164 44.688655 0.5995090
## 3 SNP3   1   3   C   A 400   0.0000 0.7925000  0.5334004 30.640305 0.9232329
## 4 SNP4   1   4   G   A 400   0.0000 0.7012500  3.1149410 41.519451 0.6287980
## 5 SNP5   1   5   A   G 400   0.0000 0.5937500 -4.0013505 38.292216 0.5178758
## 6 SNP6   1   6   C   A 400   0.0000 0.8887500 -1.6920412 17.735540 0.6878457
```

The 11 columns are: SNP (“annotation/id”), chr (“chromosome”), pos (“position”), reference and alternate alleles, the sample size N, the genotype missing rate missrate, the allele frequency of ALT allele, the score statistic SCORE of ALT allele, the variance of the score VAR, the score test P value PVAL.

2.3 Variant set tests

2.3.1 Generate random vectors Same as section 2.2.1.

2.3.2 Calculate summary statistics Same as section 2.2.2.

2.3.3 Calculating P-values G.prep and G.pval is used to calculate P-vaules for variant sets. A group definition file with no header and 6 columns (variant set id, variant chromosome, variant position, variant reference allele, variant alternative allele, weight) is required, as described in section 1.3. Here we perform variant set test in single study with an example shown as following:

```
library(StocSum)
library(GMMAT)
library(data.table)
data(example)
attach(example)
GRM.file <- system.file("extdata", "GRM.txt.bz2", package = "StocSum")
GRM <- as.matrix(read.table(GRM.file, check.names = FALSE))
nullmod <- glmmkin(disease ~ age + sex, data = pheno, kins = GRM, id = "id", family = binomial(link = "
obj <- glmmkin2randomvec(nullmod)
out.prefix <- "test.vst"
gdsfile <- system.file("extdata", "geno.gds", package = "StocSum")
G.stat(obj, geno.file = gdsfile, meta.file.prefix = out.prefix, MAF.range=c(0,0.5), miss.cutoff = 1)
```

```
## # of selected samples: 400
## # of selected variants: 100
```

```
group.file <- system.file("extdata", "SetID.withweights.txt", package = "StocSum")
obj.prep <- G.prep(out.prefix, n.files = 1, group.file = group.file, auto.flip=F)
save(obj.prep,file=paste0(out.prefix,".prepobj.Rdata"))
obj.prep <- get(load(paste0(out.prefix,".prepobj.Rdata")))
out <- G.pval(obj.prep, MAF.range = c(0,0.5), miss.cutoff = 1, method = "davies")
```

```

head(out)
```

2.3.4 Output Files

```
##   group n.variants   B.score     B.var     B.pval    E.pval
## 1  Set1         20 194.05011  84779.85 0.50512350 0.1878986
## 2  Set2         20 -82.55532 279746.84 0.87596565 0.9588988
## 3  Set3         20 184.18465 242696.40 0.70850031 0.4904610
## 4  Set4         20 296.38607  27253.92 0.07260177 0.1149147
## 5  Set5         20 446.62340  72225.19 0.09653878 0.2965374
## 6  Set6         20 260.94738 125167.13 0.46077110 0.5531424
```

It returns a data frame with the first 2 columns showing the group (variant set) name, number of variants in each group. For Burden, 3 columns will be included to show the burden test score, variance of the score, and its P value. For the efficient hybrid test, the P value column will be included in the last column.

2.4 Meta-analysis

2.4.1 single-variant meta-analysis Score test results from multiple studies can be combined in meta-analysis. The intermediate files for each study from G.stat can be used as input to the function svt.meta.

```
library(StocSum)
library(GMMAT)
library(data.table)
data(example)
attach(example)
seed <- 12345
set.seed(seed)
GRM.file <- system.file("extdata", "GRM.txt.bz2", package = "StocSum")
GRM <- as.matrix(read.table(GRM.file, check.names = FALSE))
nullmod <- glmmkin(disease ~ age + sex, data = pheno, kins = GRM, id = "id", family = binomial(link = "
if(!is.null(nullmod$P)){
    obj <- glmmkin2randomvec(nullmod)
}else{
    kinship.chol <- chol(GRM)
    obj<-glmmkin2randomvec(nullmod, Z = list(t(kinship.chol)))
}
out.prefix <- "test"
gdsfile <- system.file("extdata", "geno.gds", package = "StocSum")
G.stat(obj, geno.file = gdsfile, meta.file.prefix = out.prefix, MAF.range=c(0,0.5), miss.cutoff = 1)
```

```
## # of selected samples: 400
## # of selected variants: 100
```

```
GRM1.file <- system.file("extdata", "GRM1.txt.bz2", package = "StocSum")
GRM1 <- as.matrix(read.table(GRM1.file, check.names = FALSE))
pheno1.file <- system.file("extdata", "pheno1.txt", package = "StocSum")
pheno1 <- as.data.frame(read.table(pheno1.file, header=T,check.names = FALSE))
nullmod1 <- glmmkin(disease ~ age + sex, data = pheno1, kins = GRM1, id = "id", family = binomial(link
if(!is.null(nullmod$P)){
    obj1 <- glmmkin2randomvec(nullmod1)
}else{
    kinship1.chol <- chol(GRM1)
    obj1 <- glmmkin2randomvec(nullmod1, Z = list(t(kinship1.chol)))
}
out.prefix1 <- "test1"
gdsfile1 <- system.file("extdata", "geno1.gds", package = "StocSum")
G.stat(obj1, geno.file = gdsfile1, meta.file.prefix = out.prefix1, MAF.range=c(0,0.5), miss.cutoff = 1)
```

```
## # of selected samples: 400
## # of selected variants: 100
```

```
outMeta.prefix <- "comp.meta"
svt.meta(c("test", "test1"), n.files = rep(1, 2), outfile.prefix=outMeta.prefix, MAF.range=c(0,0.5), mi
out<-svt.pval(outMeta.prefix, n.files = 1, MAF.range=c(0,0.5), miss.cutoff = 1, auto.flip=F, nperbatch
head(out)
```

```
##    SNP chr pos ref alt   N missrate   altfreq     SCORE      VAR      PVAL
## 1 SNP1   1   1   T   A 793  0.00875 0.8909206 -4.678965 45.59656 0.4883591
## 2 SNP2   1   2   A   C 800  0.00000 0.6387500  3.681321 83.08430 0.6863065
## 3 SNP3   1   3   C   A 800  0.00000 0.7968750 -1.334301 64.40279 0.8679482
## 4 SNP4   1   4   G   A 800  0.00000 0.7606250 -4.377660 78.07962 0.6203040
## 5 SNP5   1   5   A   G 800  0.00000 0.6837500 -6.394846 78.96800 0.4717577
## 6 SNP6   1   6   C   A 800  0.00000 0.8431250 -6.875908 59.14367 0.3712796
```

2.4.2 variant set meta-analysis For variant set meta-analysis, the intermediate files for each study from G.stat can be used as input to the function G.pval. For example,

```
library(StocSum)
library(GMMAT)
library(data.table)
data(example)
attach(example)
seed <- 12345
set.seed(seed)
GRM.file <- system.file("extdata", "GRM.txt.bz2", package = "StocSum")
GRM <- as.matrix(read.table(GRM.file, check.names = FALSE))
nullmod <- glmmkin(disease ~ age + sex, data = pheno, kins = GRM, id = "id", family = binomial(link = "
if(!is.null(nullmod$P)){
    obj <- glmmkin2randomvec(nullmod)
}else{
    kinship.chol <- chol(GRM)
    obj<-glmmkin2randomvec(nullmod, Z = list(t(kinship.chol)))
}
out.prefix <- "test"
gdsfile <- system.file("extdata", "geno.gds", package = "StocSum")
G.stat(obj, geno.file = gdsfile, meta.file.prefix = out.prefix, MAF.range=c(0,0.5), miss.cutoff = 1)
```

```
## # of selected samples: 400
## # of selected variants: 100
```

```
GRM1.file <- system.file("extdata", "GRM1.txt.bz2", package = "StocSum")
GRM1 <- as.matrix(read.table(GRM1.file, check.names = FALSE))
pheno1.file <- system.file("extdata", "pheno1.txt", package = "StocSum")
pheno1 <- as.data.frame(read.table(pheno1.file, header=T,check.names = FALSE))
nullmod1 <- glmmkin(disease ~ age + sex, data = pheno1, kins = GRM1, id = "id", family = binomial(link
if(!is.null(nullmod$P)){
    obj1 <- glmmkin2randomvec(nullmod1)
}else{
    kinship1.chol <- chol(GRM1)
    obj1 <- glmmkin2randomvec(nullmod1, Z = list(t(kinship1.chol)))
}
out.prefix1 <- "test1"
gdsfile1 <- system.file("extdata", "geno1.gds", package = "StocSum")
G.stat(obj1, geno.file = gdsfile1, meta.file.prefix = out.prefix1, MAF.range=c(0,0.5), miss.cutoff = 1)
```

```
## # of selected samples: 400
## # of selected variants: 100
```

```
group.file <- system.file("extdata", "SetID.withweights.txt", package = "StocSum")
obj.prep <- G.prep(meta.files.prefix= c("test", "test1"), n.files = rep(1, 2), group.file = group.file,
outMeta.prefix <- "test.Meta"
save(obj.prep,file=paste0(outMeta.prefix,".prepobj.Rdata"))
obj.prep <- get(load(paste0(outMeta.prefix,".prepobj.Rdata")))
out <- G.pval(obj.prep, MAF.range = c(0,0.5), miss.cutoff = 1, method = "davies")
print(out)
```

```
##   group n.variants    B.score     B.var     B.pval     E.pval
## 1  Set1         20  117.15678 12767.825 0.29981350 0.16310060
## 2  Set2         20   -84.69949 34813.152 0.64986400 0.78780636
## 3  Set3         20   -27.22088 34870.852 0.88410224 0.59316701
## 4  Set4         20    72.58775  5536.723 0.32930150 0.08388703
## 5  Set5         20   265.86940 12916.629 0.01931772 0.01814263
## 6  Set6         20   -64.54065 17805.814 0.62861749 0.88701637
## 7  Set7         20 -163.42917 23016.736 0.28137832 0.59419628
## 8  Set8         20    10.58585 22577.105 0.94383411 0.96395543
## 9  Set9         20    33.37306 30067.203 0.84737876 0.76900321
```

2.5 Conditional association tests

2.5.1 Generate random vectors Same as section 2.2.1.

2.5.2 Calculate summary statistics Same as section 2.2.2.

2.5.3 Calculating P-values After the steps of generating random vectors and calculating summary statistics, Cond.svt.pval is used to calculate P-vaules.

```
library(StocSum)
library(GMMAT)
library(data.table)
data(example)
attach(example)
seed <- 12345
set.seed(seed)
GRM.file <- system.file("extdata", "GRM.txt.bz2", package = "StocSum")
GRM <- as.matrix(read.table(GRM.file, check.names = FALSE))
nullmod <- glmmkin(disease ~ age + sex, data = pheno, kins = GRM, id = "id", family = binomial(link = "
if(!is.null(nullmod$P)){
    obj <- glmmkin2randomvec(nullmod)
}else{
    kinship.chol <- chol(GRM)
    obj<-glmmkin2randomvec(nullmod, Z = list(t(kinship.chol)))
}
out.prefix <- "test"
gdsfile <- system.file("extdata", "geno.gds", package = "StocSum")
G.stat(obj, geno.file = gdsfile, meta.file.prefix = out.prefix, MAF.range=c(0,0.5), miss.cutoff = 1)
```

```
## # of selected samples: 400
## # of selected variants: 100
```

```
out <- Cond.svt.pval(out.prefix, n.files = 1, tagChr = 1, StartPos = 1, EndPos = 100, tagPos = 82, MAF.
head(out)
```

```
##    SNP chr pos ref alt   N missrate altfreq      SCORE       VAR      PVAL
## 1 SNP2   1   2   A   C 400        0 0.50000  3.5457182 46.209665 0.6019487
## 2 SNP3   1   3   C   A 400        0 0.79250  0.2460479 27.973531 0.9628953
## 3 SNP4   1   4   G   A 400        0 0.70125  4.4627558 41.526950 0.4886050
## 4 SNP5   1   5   A   G 400        0 0.59375 -2.6813778 41.105056 0.6757823
## 5 SNP6   1   6   C   A 400        0 0.88875 -2.1134340 18.581522 0.6239327
## 6 SNP7   1   7   C   A 400        0 0.96875  4.3930752  5.600377 0.0634036
##           cor
## 1 0.001958348
## 2 0.022923247
## 3 0.087793059
## 4 0.086445300
## 5 0.041066100
## 6 0.014463935
```

2.6 Gene-environment tests

2.6.1 Generate random vectors Same as section 2.2.1.

2.6.2 Calculate summary statistics Different from G.state, the "GE.stat" needs the arguments "interaction" and "intearaction.covariates" as input. An example is as following:

```
library(StocSum)
library(GMMAT)
library(data.table)
data(example)
attach(example)
seed <- 12345
set.seed(seed)
GRM.file <- system.file("extdata", "GRM.txt.bz2", package = "StocSum")
GRM <- as.matrix(read.table(GRM.file, check.names = FALSE))
nullmod <- glmmkin(disease ~ age + sex, data = pheno, kins = GRM, id = "id", family = binomial(link = "
if(!is.null(nullmod$P)){
    obj <- glmmkin2randomvec(nullmod)
}else{
    kinship.chol <- chol(GRM)
    obj<-glmmkin2randomvec(nullmod, Z = list(t(kinship.chol)))
}
out.prefix <- "test.GE"
gdsfile <- system.file("extdata", "geno.gds", package = "StocSum")
GE.stat(obj, interaction='sex', geno.file = gdsfile, meta.file.prefix = out.prefix)
```

```
## # of selected samples: 400
## # of selected variants: 100
## # of selected variants: 100
```

2.6.3 Calculating P-values When the intermediate files are generated by "GE.stat", the function "GE.svt.pval" can be used to calculate P-values for each variants. An example is as following:

```
out.file<-paste0(out.prefix,".out")
GE.svt.pval(meta.files.prefix = out.prefix, out.file, n.files = 1, n.pheno = 1, interaction='sex', MAF.
system("head test.GE.out")
```

2.7 LD score regression

2.7.1 Generate random vectors To run StocSum.LDSC.R, the user needs to generate the random vectors from multivariate normal distrubution with mean 0 and covariance matrix P=I-1(1'1)^(-1)1'.

```
library(StocSum)
library(GMMAT)
library(data.table)
data(example)
attach(example)
seed <- 12345
set.seed(seed)
GRM.file <- system.file("extdata", "GRM.txt.bz2", package = "StocSum")
GRM <- as.matrix(read.table(GRM.file, check.names = FALSE))
```

```
nullmod <- glmmkin(disease ~ age + sex, data = pheno, kins = GRM, id = "id", family = binomial(link = "
obj <- LDSC.glmmkin2randomvec(nullmod)
```

The function LDSC.glmmkin2randomvec returns a list. The element random.vectors stores the generated random vectors. The remaining elements theta, scaled.residuals, and id_include are inherited from the null model generated in 2.1.

2.7.2 Calculate summary statistics Same function G.stat as in secton 2.2.2.

2.7.3 Calculating P-values The function LDSC.win is used to calculate LD scores. The argument "wind.b" defines the window size in kilobases to estimate LD Scores.

```
out.prefix <- "test"
gdsfile <- system.file("extdata", "geno.gds", package = "GMMAT")
G.stat(obj, geno.file = gdsfile, meta.file.prefix = out.prefix, MAF.range=c(0,0.5), miss.cutoff = 1)
```

```
## # of selected samples: 400
## # of selected variants: 100
```

```
out<-LDSC.win(out.prefix, use.minor.allele = FALSE, auto.flip = FALSE, wind.b = 1000000, nperbatch = 10
```

2.7.4 Output The 6 columns are: chr ("chromosome"), pos ("position"), the sample size N, the genotype missing rate missrate, the allele frequency of ALT allele, LD Scores.

```
head(out)
```

```
##   chr pos   N missrate   altfreq   LDscore
## 1   1   1 393   0.0175 0.9745547 1.6087206
## 2   1   2 400   0.0000 0.5000000 1.3015771
## 3   1   3 400   0.0000 0.7925000 1.3612845
## 4   1   4 400   0.0000 0.7012500 1.7007805
## 5   1   5 400   0.0000 0.5937500 1.4388094
## 6   1   6 400   0.0000 0.8887500 0.8787695
```