User Manual for

MRBIGR

Mendelian Randomization-Based Inference of Genetic Regulation

Last updated in Sept. 2021

Contents

1 Introduction

MRBIGR is a multifunctional toolkit for pre-GWAS, GWAS and post-GWAS of both traditional and multi-omics data. MRBIGR provides all the components needed to build a complete GWAS pipeline, and integrated with rich post-GWAS analysis tools such as QTL annotation and haplotype analysis. In particular, Mendelian randomization (MR) analysis, MR-based network construction, module identification and gene ontology analysis are proposed for further genetic regulation studies. Additionally, it also produces rich plots for visualization of the analysis results and other formatted data.

2 Installation

2.1 Download

```
git clone https://github.com/CrazyHsu/MRBIGR.git
```

2.2 Activate a new conda environment

```
conda create -n mrbigr python=3.9 -y
conda activate mrbigr
```

2.3 Build and install

```
cd MRBIGR
python setup.py build
python setup.py install
```

2.4 Install dependencies

```
pip install pyranges
conda install -y -c conda-forge r-rcppeigen r-xml r-rsqlite r-europepmc r=3.6 rpy2
Rscript -e 'install.packages(c("data.table", "ggplot2", "ggsignif", "ggrepel","Matrix", "igraph", "network", "GGally", "sna",  "europepmc","ggraph"), repos="https://cloud.r-project.org")'
Rscript -e 'if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager", repos="https://cloud.r-project.org");BiocManager::install(c("AnnotationForge","clusterProfiler"))
Rscript -e 'install.packages("bigsnpr", dependence=T, repos="https://cloud.r-project.org")'
```

2.5 Set environment variables

```
echo "export PATH=`pwd`/utils:\$PATH" >> ~/.bashrc
source ~/.bashrc
```

3 Quick Start

MRBIGR consists of seven analysis modules: genotype based analysis module (`geno`), phenotype based analysis module (`pheno`), GWAS and QTL related analysis module (`gwas`), Mendelian randomization analysis module (`mr`), MR-based network construction module (`net`), gene ontology analysis module (`go`) and data visulaization module (`plot`). Each module can be invoked via a subcommand, and each module also provides several functions which can be called with corresponding parameters. In this part, only several commonly used functions in each module are shown as examples. For more detailed instructions, please refer to the tutorial section.

3.1 Genotype data process

Suppose we have a set of plink-bed format genotype data named *`geno.bed`*/*`geno.bim`*/*`geno.fam`*. First, quality control of the original genotype data should be performed through the below command:

```
MRBIGR.py geno -qc -g geno -o geno_qc -maf 0.05 -mis 0.2 -mind 0.2
```

The subcommand `geno` invokes the genotype analysis module; parameter `-qc` calls the quality control function; `-g` is the plink-bed format input genotype data; `-o` is the output genotype data prefix; `-maf` is the MAF for a SNP to be kept; `-mis` is the maximum proportion of missing values for a SNP to be kept; `-mind` is the maximum proportion of missing values for a sample to be kept. After this step, the QC-filtered genotype data *`geno_qc.bed`*/*`geno_qc.bim`*/*`geno_qc.fam`* would be generated. Then, dimensionality reduction of the original genotype data could be performed using the below command:

```
MRBIGR.py geno -clump -g geno_qc -o geno
```

The subcommand `geno` invokes the genotype analysis module; parameter `–clump` calls the genotype clumping function to keep only one representative SNP per region of LD; `-g` is the plink-bed format input genotype data, `-o` is the output genotype data prefix (*`suffix _clump`* will be added automatically for the output files). After this step, the clumped genotype data *`geno_clump.bed`*/*`geno_clump.bim`*/*`geno_clump.fam`* would be generated. Then, take the dimensional reduced genotype data as input to perform principal component analysis using the below command:

```
MRBIGR.py geno -pca -g geno_clump -o geno
```

The subcommand `geno` invokes the genotype analysis module; parameter `–pca` calls the principal component analysis function; `-g` is the plink-bed format input genotype data; `-o` is the output genotype data prefix (`suffix_pca` will be added automatically for the output files). After this step, an output file named *`geno_pca.csv`* would be generated in the working directory.

3.2 Phenotype data process

Suppose we have a CSV format phenotype file named pheno.csv, the first column and the first row should be the names of samples and traits, respectively. First, quality control of the original phenotype data should be performed through the below command:

```
MRBIGR.py pheno –qc -p pheno.csv -o pheno_qc -mis 0.5 -val 0.1 –rout
```

The subcommand `pheno` invokes the phenotype analysis module; parameter `–qc` calls the quality control function; `-p` is the input phenotype matrix; `-o` is the prefix of output file; `-mis` is the missing rate cutoff; `-val` is the small value cutoff; `-rout` means outlier removal of phenotypic values with the default method. After this step, a QC-filtered phenotype data named *`pheno_qc.phe.csv`* would be generated. Then, take this file as input, perform missing phenotypic value imputation through the below command:

```
MRBIGR.py pheno --imput -p pheno_qc.phe.csv -o pheno_imput
```

The subcommand `pheno` invokes the phenotype analysis module; parameter `–imput` calls the phenotype imputation function; `-p` is the input phenotype matrix; `-o` is the prefix of output file. After this step, a phenotype file named *`pheno_imput.phe.csv`* with no NA value would be generated. Then, take this file as input to perform phenotypic value normalization using the below command:

```
MRBIGR.py pheno -scale -p pheno_imput.phe.csv –o pheno_norm -boxcox -minmax
```

The subcommand `pheno` invokes the phenotype analysis module; parameter `–scale` calls the phenotype scaling/normalization/standardization/transformation function; `-p` is the input phenotype matrix; `-o` is the prefix of output file, `-boxcox` and `-minmax` means perform both box-cox normalization and min-max scaling of the input data. After this step, a phenotype file named pheno_norm.phe.csv with normalized and scaled phenotypic values would be generated.

In some cases, phenotype data are collected from different environment or years, which should be merged through appropriate algorithm such as mean-values or BLUP before further analysis. Here is an example, take one trait from different environment as input, the below command line will merge all the columns using BLUP algorithm:

```
MRBIGR.py pheno -merge -p trait1.csv -o trait1_blup -mm blup
```

The subcommand `pheno` invokes the phenotype analysis module; parameter `-merge` calls the phenotype merge function; `-p` is the input phenotype matrix for a trait in CSV format; `-o` is the prefix of output file; `-mm` is the merge method. After this step, a phenotype file named *`trait1_blup.phe.csv`* with BLUP merged phenotypic values would be generated.

3.3 GWAS and QTL detection

Take the processed plink-bed format genotype data and CSV format phenotype data as inputs, GWAS can be performed through the below command:

```
MRBIGR.py gwas -gwas -model lmm -thread 12 -g geno_qc -p pheno_norm.csv
```

The subcommand `gwas` invokes the GWAS and QTL analysis module; parameter `-gwas` calls the GWAS function; `-model` is the model to fit, with linear mixed model (`lmm`) by default; `-thread` is the thread number to run the command; `-g` is the plink-bed format input genotype data; `-p` is the CSV format phenotype data. After this step, an output directory would be generated with the GWAS result files named *`trait_name.assoc.txt`* in it. Then, QTL can be determined using the below command based on the GWAS results:

```
MRBIGR.py gwas -qtl -g geno_qc -thread 6 -i output -o qtl_output -p1 1e-5 -p2 1e-3 -p2n 5
```

The subcommand `gwas` invokes the GWAS and QTL analysis module; parameter `-qtl` calls the QTL detection function; `-thread` is the thread number to run the command; `-i` is the GWAS result directory; `-o` is the output file prefix; `-p1` is the significance threshold for index SNPs used to determine QTLs; `-p2` is the secondary significance threshold for clumped SNPs used to determine the reliability of QTLs; -p2n is secondary significance SNP number in a QTL. After this step, an output file named *`qtl_output.qtl_res.csv`* would be generated. If you want to generate manhattan-plots and QQ-plots for visualization of the GWAS results, the below command should be helpful:

```
MRBIGR.py gwas -visual -thread 12 -i output -o vis_output -multi -q qtl_output.qtl_res.csv
```

The subcommand `gwas` invokes the GWAS and QTL analysis module; parameter `-visual` calls the visualization function; `-thread` is the thread number to run the command; `-i` is the GWAS result directory; `-o` is the output directory for the generated plots files; parameters `-multi` and `-q` are optional, if `-multi` is set to generate a multi-traits manhattan-plot, `-q` which is the QTL result file also need to be set.

3.4 Mendelian randomization analysis

Take the plink-bed format genotype data, CSV format exposure data, CSV format exposure QTL data and CSV format outcome data as inputs, Mendelian randomization analysis can be performed through the below command:

```
MRBIGR.py mr -g geno_qc -exposure exposure.csv -qtl exposure_qtl.csv -outcome outcome.csv -mlm -thread 12 -o mr_out
```

The subcommand `mr` invokes the Mendelian randomization analysis module; parameter `-exposure` is the CSV format exposure data; `-outcome` is the CSV format outcome data; `-qtl` is the CSV format exposure QTL data; `-thread` is the thread number to run the command; -`g` is the plink-bed format input genotype data; `-mlm` represents perform Mendelian randomization analysis through mixed linear model; `-o` is the prefix of output file. After this step, a MR result file named *`mr_out.MR.csv`* would be generated.

It also can take the CSV format population gene expression data and CSV format gene QTL data as inputs, pairwise genes Mendelian randomization analysis can be performed through the below command:

```
MRBIGR.py mr -g geno_qc -gene_exp gene_exp.csv -pairwise -mlm -qtl gene_qtl.csv -thread 12  -o pairwise_mr_out
```

The subcommand `mr` invokes the Mendelian randomization analysis module; parameter `-gene_exp` is the CSV format population gene expression data; `-qtl` is the CSV format gene QTL data; `-thread` is the thread number to run the command; `-g` is the plink-bed format input genotype data; `-mlm` represents perform Mendelian randomization analysis through mixed linear model; `-o` is the prefix of output file. After this step, a MR result file named *`pairwise_mr_out.MR.csv`* would be generated. If you want to perform Mendelian randomization analysis between transcription factor and genes it targets, the below command should be helpful:

```
MRBIGR.py mr -g geno_qc -gene_exp gene_exp.csv -tf tf_genefunc.txt -target target_genefunc.txt -mlm -qtl tf_qtl.csv -threads 12 -o tf_mr_out
```

The subcommand mr invokes the Mendelian randomization analysis module; parameter `-gene_exp` is the CSV format population gene expression data; `-qtl` is the CSV format transcription factor QTL data; `-tf` is the transcription factor annotation file; `-target` is the annotation files for genes targeted by transcription factors; `-thread` is the thread number to run the command; `-g` is the plink-bed format input genotype data; `-mlm` represents perform Mendelian randomization analysis through mixed linear model; `-o` is the prefix of output file. After this step, a MR result file named *`tf_mr_out.MR.csv`* would be generated.

3.5 MR-based network construction

Take the CSV format Mendelian randomization analysis data as input, Mendelian randomization based network analysis can be performed through the below command:

```
MRBIGR.py net -mr pairwise_mr_out.MR.csv -plot -o net_out
```

The subcommand net invokes the Mendelian randomization based network analysis module; parameter `-mr` is the CSV format Mendelian randomization analysis data; `-plot` represents plot network for each identified network module ;`-o` is the prefix of output file. After this step, network edgelist file *`net_out.edge_list`*, ClusterONE software result *`net_out.clusterone.result.csv`*, network module *`plot net_out.module*.pdf`* and final network module file *`net_out.module.csv`* would be generated.

3.6 Gene ontology analysis of network modules

Take the CSV format network module data, Tabular format gene ontology annotation of each gene data and Tabular format gene annotation data as inputs, gene ontology enrichment analysis of each module can be performed through the below command:

```
MRBIGR.py go -gene_lists net_out.module.csv -go_anno go_anno.txt -
gene_info gene_anno.txt -o go_out
```

The subcommand `go` invokes the gene ontology enrichment analysis module; parameter `-gene_lists` is the CSV format network module data; `-go` is the tabular format gene ontology annotation of each gene data; `-gene_info` is the tabular format gene annotation data; `-o` is the prefix of output file. After this step, gene ontology enrichment analysis result *`go_out.GO.csv`*, visualized results of functional enrichment results *`go_out.BP.dotplot.pdf`*, *`go_out.MF.dotplot.pdf`* and *`go_out.CC.dotplot.pdf`* would be generated.

4 Tutorial

In this part, the implementation of each module/function/parameter in MRBIGR will be introduced in detail, and several real-data results will also be shown to help users better understand the functions of MRBIGR.

4.1 Genotype data based analysis

The genotype data analysis module takes plink-bed format genotype data as input, can be invoked through the subcommand `geno,` which includes 9 functions: `-convert`, -`qc,` `-imput`, `-clump`, `-kinship`, `-pca`, `-tsne`, `-tree` and `-anno.` `-convert` is used for genotype data format conversion; `-qc` is used for quality control of genotype data; `-imput` introduced several simple methods for fast imputation of genotype data; `-clump` is used to keep only one representative SNP per region of LD based on a clumping method, to reduce the dimensionality of the original genotype data; `-kinship` is used to calculate the kinship matrix from genotype data; `-pca` is used to perform principle component analysis from genotype data; `-tsne` is used to perform t-SNE analysis from genotype data; `-tree` is used to build a ML-tree from genotype data; `-anno` is used to annotate vcf-format genotype data.

4.1.1 Format conversion of genotype data

MRBIGR supports the conversion of Hapmap/VCF format genotype data to plink-bed format, as well as the conversion of plink-bed format to VCF format. This function can be called through the parameter `-convert`. The command line is as follows:

```
# Hapmap to plink-bed
MRBIGR.py geno -convert -hapmap test.hapmap -o test_hmp
# VCF to plink-bed
MRBIGR.py geno -convert -vcf test.vcf -o test_vcf
# plink-bed to VCF
MRBIGR.py geno –g geno –o geno.vcf
```

The subcommand `geno` invokes the genotype data analysis module; parameter `–convert` calls the data format conversion function; `-hapmap` and `-vcf` are the correspond format file name; `-g` is the prefix of plink-bed format input data, `-o` is the name of output data.

4.1.2 Quality control of genotype data

The raw genotype data may contain SNPs with low values of MAF (minor allele frequency), high level of missing rate, as well as samples with extreme high missing rate. These data should be filtered before downstream analysis. The `geno` module provides a QC-based genotype data filter function which can be called through parameter `–qc`. The command line is as follows:

```
MRBIGR.py geno -qc -g geno -o geno_qc -maf 0.05 -mis 0.2 -mind 0.2
```

The subcommand `geno` invokes the genotype analysis module; parameter `–qc` calls the quality control function; `–g` is the plink-bed format input genotype data; `-o` is the output genotype data prefix; `-maf` is the MAF for a SNP to be kept; `-mis` is the maximum proportion of missing values for a SNP to be kept; `-mind` is the maximum proportion of missing values for a sample to be kept. The parameters `–maf`, -`mis` and `–mind` have default values and are not mandatory.

4.1.3 Fast imputation of missing genotype values

MRBIGR provides several fast and simple methods to impute the missing values in the input genotype data, either “`random`” (sampling according to allele frequencies), “`mean0`” (rounded mean), “`mean2`” (rounded mean to 2 decimal places), “`mode`” (most frequent call). It should be noted that these methods are just for fast imputation of test dataset, more accurate imputation of genotype data should be performed using other professional imputation tools such as Beagle. The command line is as follows:

```
MRBIGR.py geno -imput -method mode -g geno_qc -o geno
```

The subcommand `geno` invokes the genotype analysis module; parameter `-imput` calls the genotype imputation function; `-method` is the imputation method; `-g` is the plink-bed format input genotype data, `-o` is the output genotype data prefix (*suffix_imput* will be added automatically for the output files).

4.1.4 Dimensionality reduction of genotype data through SNP clumping

SNP clumping is used to keep only one representative SNP per region of LD, which can be used to reduce dimensionality of the genotype data. This method is proposed in R package bigsnpr, it is similar to the LD-based SNP pruning method in PLINK, but SNP clumping has the advantage of only the SNP with the highest MAF is kept in a LD region and more genetic variation can be kept. The command line is as follows:

```
MRBIGR.py geno -clump -g geno_qc -o geno
```

The subcommand `geno` invokes the genotype analysis module; parameter `-clump` calls the genotype clumping function to keep only one representative SNP per region of LD; `-g` is the plink-bed format input genotype data, `-o` is the output genotype data prefix (*suffix _clump* will be added automatically for the output files).

4.1.5 Calculation of kinship matrix from genotype data

Pairwise kinship coefficients calculation is used as a measurement of genetic similarity between individuals. MRBIGR can generate a kinship/relatedness matrix by the below command:

```
MRBIGR.py geno -kinship -g geno_clump -o kinship
```

The subcommand `geno` invokes the genotype analysis module; parameter `-kinship` calls the kinship calculation function; `-g` is the plink-bed format input genotype data, `-o` is the output file prefix. The output file `kinship.cXX.txt` could be found in a directory named output.

4.1.6 Principle component analysis from genotype data

Principal components analysis (PCA) is commonly applied to population structure inference and dimension reduction of the data. Top PCs calculated from the genotype data can reflect population structure among the sample individuals. MRBIGR can perform PCA analysis by the below command:

```
MRBIGR.py geno -pca -g geno_clump -o geno -dim 10
```

The subcommand `geno` invokes the genotype analysis module; parameter `-pca` calls the PCA analysis function; `-g` is the plink-bed format input genotype data, `-o` is the output file prefix, `-dim` is the dimensionality or PCs for the output data, with a default value 10. Then, a CSV format output file named `geno_pc.csv` would be generated.

4.1.7 t-SNE analysis from genotype data

t-SNE (t-Distributed Stochastic Neighbor Embedding) is a machine learning algorithm for dimensional reduction. As a nonlinear dimensional reduction algorithm, t-SNE performs better than PCA, and is suitable for visualization of population structure. MRBIGR first use PCA to reduce the data to 50 dimensions, and then use t-SNE algorithm to further reduced the data to 2 or 3 dimensions. The command line is as follows:

```
MRBIGR.py geno -tsne -g geno_clump -o geno -dim 3
```

The subcommand `geno` invokes the genotype analysis module; parameter `-tsne` calls the t-SNE analysis function; `-g` is the plink-bed format input genotype data, `-o` is the output file prefix, `-dim` is the dimensionality for the output data, usually 2 or 3, with a default value 3. Then, a CSV format output file named `geno_tsne.csv` would be generated.

4.1.8 Phylogenetic analysis from genotype data

MRBIGR introduces a one-step phylogenetic analysis function from genotype data, which facilitate the visualization of relatedness of individuals and population structure. A nwk format maximum likelihood (ML) tree can be generated follow this command:

```
MRBIGR.py geno -tree -g geno_clump -o geno_tree
```

The subcommand `geno` invokes the genotype analysis module; parameter `-tree` calls the phylogenetic analysis function; `-g` is the plink-bed format input genotype data, `-o` is the output file prefix. Then, a nwk format output file named `geno_tree.tree.nwk` and some other related files geno_tree.* would be generated.

4.1.9 Functional annotation of genotype data

This function in MRBIGR is relied on ANNOVAR, an efficient software tool to utilize update-to-date information to functionally annotate genetic variants. Only `vcf-format` genotype data is supported as input, plink-bed format genotype data should be converted to VCF format using the `-convert` function. If annotation is performed for the first time, an annotation database should be built by adding the parameters `-gtf` and `-fa`. The command line is as follows:

```
MRBIGR.py geno -anno -vcf test.vcf -o testvcf_anno -db ZmB73 -dbdir ./ref -gtf ./ref/Zea_mays.Zm-B73-REFERENCE-NAM-5.0.51.chr.gtf -fa ./ref/Zea_mays.Zm-B73-REFERENCE-NAM-5.0.dna.toplevel.fa
```

The subcommand `geno` invokes the genotype analysis module; parameter `-anno` calls the annotation function; `-vcf` is the `vcf-format` input genotype data, `-o` is the output file prefix; `-db` is the output database name; `-dbdir` is the output database directory; `-fa` is the reference genome sequences file in FASTA format; `-gtf` is the reference gene annotation file in GTF format. Then, MRBIGR would generate an annotation database and perform functional annotate for genetic variants automatically. The output annotation result files include *testvcf_anno.ZmB73_multianno.vcf*, *testvcf_anno.ZmB73_multianno.bed*, and *testvcf_anno.ZmB73_largeEffect.bed*. If the annotation database has been built before, the parameters `-gtf` and `-fa` are no longer need, and the parameters `-db` and `-dbdir` are the input database name and database directory, respectively. In this case, the command line is as follows:

```
MRBIGR.py geno -anno -vcf test2.vcf -o testvcf2_anno -db ZmB73 -dbdir ./ref -gtf ./ref/Zea_mays.Zm-B73-REFERENCE-NAM-5.0.51.chr.gtf
```

4.2 Phenotype data based analysis

The phenotype data analysis module takes a CSV format phenotype file with the first column and the first row should be the names of samples and traits as input, can be invoked through the subcommand `pheno`, which includes 5 functions: `-qc`, `-imput`, `-scale`, `-correct`, and `-merge`. `-qc` is used for quality control of phenotype data; `-imput` is used for imputation of missing values in phenotype data; `-scale` is used for data scaling, normalization, standardization or transformation of phenotype data; `-correct` is used for population structure based phenotype data correction; `-merge` is used to merge a trait from different environment or years using either the methods of mean values, BLUP or BLUE.

4.2.1 Quality control of phenotype data

Quality control of the original phenotype data include transposition of the original phenotype matrix if needed, removal of outlier data points based on z-score or boxplot methods, filtering traits with high level of missing rate and low average values. In this function, either of the parameters `-tr`, `-rout`, `-mis` and `-val` is optional. If you want to perform quality control use all the default parameters, the command line looks like this

```
MRBIGR.py pheno -qc -p pheno.csv -o pheno_qc -rout -mis -val
```

The subcommand `pheno` invokes the phenotype analysis module; parameter `-qc` calls the quality control function; `-p` is the input phenotype matrix; `-o` is the prefix of output file; `-mis` is the missing rate cutoff with default value 0.5; `-val` is the small value cutoff with default value 0.1; `-rout` means outlier removal of phenotypic values with the default method z-score. If you want to perform quality control use personalized parameters, the command line could be as follows:

```
# transposition of the original phenotype matrix
MRBIGR.py pheno -qc -tr pheno.csv -o pheno_tr
# reset missing rate cutoff, skip small value filter and outlier removal
MRBIGR.py pheno -qc -p pheno.csv -o pheno_qc -mis 0.1
# skip small value filter and set outlier removal method to boxplot
MRBIGR.py pheno -qc -p pheno.csv -o pheno_qc -rout boxplot -mis
```

4.2.2 Imputation of missing phenotype values

There are three different phenotype imputation methods provided by MRBIGR: `mean`, `median`, `most_frequent`. The missing phenotype values in input file should be defined as NA, and the command line is as follows:

```
MRBIGR.py pheno --imput -p pheno_qc.phe.csv -o pheno_imput -method most_frequent
```

The subcommand `pheno` invokes the phenotype analysis module; parameter `-imput` calls the phenotype imputation function; `-p` is the input phenotype matrix; `-o` is the prefix of output file; `-method` is the imputation method. After this step, a phenotype file named *`pheno_imput.phe.csv`* with no missing value would be generated.

4.2.3 Scaling and normalization of phenotype data

MRBIGR provides multiple commonly used phenotype data scaling and normalization methods, such as logarithmization, z-score standarlization, box-cox normalization, normal quantile normalization. These scaling and normalization methods are applicable to different types of phenotype data, like agronomic traits, transcripts expression data, metabolites intensity data. The command line is as follows:

```
MRBIGR.py pheno -scale -p pheno_imput.phe.csv -o pheno_norm -boxcox -minmax
```

The subcommand `pheno` invokes the phenotype analysis module; parameter `-scale` calls the phenotype scaling/normalization/standardization/transformation function; `-p` is the input phenotype matrix; `-o` is the prefix of output file; `-boxcox` and `-minmax` means use both Box-Cox and Min-Max methods to transform the data. Box-Cox transformation is used to transform each trait to meet normality assumptions, a lambda is calculated per trait and used to transform each trait, while Min-Max scaling is used to scale the values to 0-1. Other optional methods are `-log2`, `-log10`, `-ln`, `-qqnorm`, `-zscore`, and `-robust`.

4.2.4 Correction of phenotype data base on population structure

Since population structure may lead to phenotype data distributed in different levels, in some cases, correction of phenotype values to the same level based on population structure may help downstream analysis. The `-correct` function in MRBIGR use a PCA file, which can be generated through genotype data, to perform phenotype data correction. The command line is as follows:

```
MRBIGR.py pheno -correct -p pheno_imput.phe.csv -o pheno_correct -pca geno_pca.csv
```

The subcommand `pheno` invokes the phenotype analysis module; parameter `-correct` calls the phenotype correct function; `-p` is the input phenotype matrix; `-o` is the output file prefix; `-pca` is the PCA result file generated by genotype data through the command `geno -pca`.

4.2.5 Merge the phenotype values from different environment

Phenotype data form different environment or years need to be merged before downstream analysis, commonly used phenotype data merge algorithms are mean-value, BLUP (best linear unbiased prediction) and BLUE (best linear unbiased estimation). MRBIGR provides all three methods for this purpose, which can be called and selected from parameters `-merge` and `-mm`. It is noted that only one trait is accepted in an input file. The command line is as follows:

```
MRBIGR.py pheno -merge -p trait1.csv -o trait1_blup -mm blup
```

The subcommand `pheno` invokes the phenotype analysis module; parameter `-merge` calls the phenotype merge function; `-p` is the input phenotype matrix for a trait in CSV format; `-o` is the prefix of output file; `-mm` is the merge method. After this step, a phenotype file named *`trait1_blup.phe.csv`* with BLUP merged phenotypic values would be generated.

4.3 GWAS and QTL related analysis

The GWAS and QTL related analysis module can be utilized for GWAS, QTL detection and annotation, as well as peak or gene based haplotype analysis, this module can be invoked through the subcommand `gwas`, which includes 6 functions: `-gwas`, `-qtl`, `-anno`, `-visual`, `-peaktest`, and `-`

`genetest`. `-gwas` is used to perform GWAS, `-qtl` is used to detect QTLs from GWAS results, `-anno` is used for annotation of QTL regions, `-visual` is used for the visualization of GWAS results, `-peaktest` is used for peak based haplotype test, `-genetest` is used for gene based haplotype test.

4.3.1 Parallel GWAS

MRBIGR support parallel GWAS for multiple traits. Take the processed plink-bed format genotype data and CSV format phenotype data as inputs, GWAS can be performed through the below command:

```
MRBIGR.py gwas -gwas -model lmm -thread 12 -g geno_qc -p pheno_norm.csv
```

The subcommand `gwas` invokes the GWAS and QTL analysis module; parameter `-gwas` calls the GWAS function which utilizes GEMMA to perform GWAS; `-model` is the model to fit, with either linear mixed model (`lmm`) or linear model (`lm`) as option; `-thread` is the thread number to run the command; `-g` is the plink-bed format input genotype data; `-p` is the CSV format phenotype data. After this step, an output directory would be generated with the GWAS result files named trait_name.assoc.txt in it.

4.3.2 Parallel QTL detection based on GWAS results

MRBIGR introduces the clumped method implemented in PLINK v1.9 to automatically detect and optimize QTLs based on the original GWAS results. In detail, a stricter P-value threshold `-p1` is set to uncover the significantly associated SNPs; then, for each significantly associated SNP, if the other SNPs within a 500 kb distance have P-values smaller than the second looser P-value threshold `-p2`, and have r2 values greater than 0.2 with the index SNP, as well as the number of such SNPs surpass the SNP number threshold `-p2n`, then the region is regarded as a QTL; finally, all overlapping QTLs are merged to generate the final QTL set, while the SNP with the smallest P-value in a QTL is defined as a lead SNP.The command line is as follows:

```
MRBIGR.py gwas -qtl -g geno_qc -thread 6 -i output -o qtl_output -p1 1e-5 -p2 1e-3 -p2n 5
```

The subcommand `gwas` invokes the GWAS and QTL analysis module; parameter `-qtl` calls the QTL detection function; `-thread` is the thread number to run the command; `-i` is the GWAS result directory; `-o` is the output file prefix; `-p1` is the significance threshold for index SNPs used to determine QTLs; `-p2` is the secondary significance threshold for clumped SNPs used to determine the reliability of QTLs; `-p2n` is secondary significance SNP number in a QTL. After this step, an output file named *`qtl_output.qtl_res.csv`* would be generated.

4.3.3 Annotation of QTL regions

The QTL result file could be annotated use a four columns gene annotation file in TSV format, which looks like this:

```
geneid  aliased position    function
Zm00001eb015280 Zm00001eb015280 1:52319290-52320913:+   Zm00001eb015280
Zm00001eb000610 Zm00001eb000610 1:2555438-2555822:+     Zm00001eb000610
Zm00001eb033210 Zm00001eb033210 1:184900367-184903962:+ Zm00001eb033210
```

This file could be generated from a standard GTF format gene annotation file through the follow command:

```
grep -v '#' input.gtf |awk '{if($3=="gene"){print $0}}' |sed 's/ /\t/g'|sed 's/"//g'|sed 's/;//g'|awk '{print $10"\t"$10"\t"$1":"$4"-"$5":"$7"\t"$10}'|sed '1igeneid\taliased\tposition\tfunction' >gene.anno.tsv
```

Then, take this file as input, QTL annotation could be performed use follow the command:

```
MRBIGR.py gwas -anno -q qtl_output.qtl_res.csv -a gene.anno.tsv -o anno_output
```

The subcommand `gwas` invokes the GWAS and QTL analysis module; parameter `-anno` calls the QTL annotation function; `-q` is the input QTL result file for annotation file; `-o` is the prefix of output file. After this step, an output file named *`anno_output.qtl_anno.csv`* would be generate.

4.3.4 Visualization of GWAS results

Manhattan-plots and QQ-plots can be generated for the visualization purpose of the GWAS results. The command line is as follows:

```
MRBIGR.py gwas -anno -q qtl_output.qtl_res.csv -a gene.anno.tsv -o anno_output
```

or

```
python MRBIGR.py gwas -visual -i output -o vis_output -multi -q qtl_output.qtl_res.csv -thread 12
```

The subcommand `gwas` invokes the GWAS and QTL analysis module; parameter `-visual` calls the visualization function; `-i` is the GWAS result directory; `-o` is the output directory prefix; `-multi` is optional, with a multi-trait Manhattan-plot would be generate when set; `-q` is the input QTL result file, it only need to be set when `-multi` is set; `-thread` is the thread number to run the command.

4.3.5 Statistical test of lead SNP

A simple approach to establish the relationship of phenotype distribution and QTL haplotype type is to use genotype of the lead SNP to represent the haplotype type of the QTL region, then, phenotype values distribution under different genotype of the lead SNP could be displayed, and student's test could be performed. This simple statistical test can be performed in MRBIGR through the command below:

```
python MRBIGR.py gwas -peaktest -o peaktest_output -p pheno.csv -
g geno_output_clump -q qtl_output.qtl_res.csv
```

The subcommand `gwas` invokes the GWAS and QTL analysis module; parameter `-peaktest` calls the peak test function; `-o` is the directory of output files; `-p` is the input phenotype data; `-g` is the input genotype data; `-q` is the input QTL result file. Then, genotype information of each sample and phenotype distribution plot would be generated in the output directory.

4.3.6 Statistical test of genes

Another approach to establish the relationship of phenotype distribution and QTL haplotype is to test the phenotype distribution and haplotype type for each gene in the QTL, to discover potential casual gene and variants. For a QTL region, MRBIGR uses large effect variants in each gene to build gene haplotypes, and perform gene based haplotype test. Welch's test was used for a two-group haplotype test and a Tukey's test was used for a multiple group haplotype test as described in Yano *et al*, 2016. The command line is as follows:

```
python MRBIGR.py gwas -genetest -f1 anno_output.qtl_anno.csv -
f2 testvcf_anno.ZmB73_multianno.bed -vcf test.vcf -p pheno.csv -
o genetest_output
```

The subcommand `gwas` invokes the GWAS and QTL analysis module; parameter `-genetest` calls the gene test function; `-f1` is the input QTL annotation file generated by `gwas -anno` command; `-f2` is the input genotype annotation file generated by `geno -anno` command; `-p` is the input phenotype file; `-o` is the directory of output files. Then, related gene haplotype information of each sample and phenotype distribution plot would be generated in the output directory.

4.4 Mendelian randomization analysis of multi-omics data

The Mendelian randomization analysis module can be utilized for perform Mendelian randomization analysis between different omics data, this module can be invoked through the subcommand `mr`, which includes 3 modes according to parameters. The first mode provides input omics data through the `-exposure` and `-outcome` parameters, and performs Mendelian randomization analysis between `-exposure` data and `-outcome` data; the second mode provides transcriptome expression data through the `-gene_exp` parameter, and perform Mendelian randomization analysis between gene pairs through the `-pairwise` parameter; the third mode provides transcriptome expression data through the `-gene_exp` parameter, the `-tf` parameter provides the annotation information of the transcription factor, and -target provides the transcription factor target gene annotation information, and then perform Mendelian randomization analysis between transcription factors and target genes. There are two calculation models for Mendelian randomization, linear model and mixed linear model, respectively, specified by the parameters `-lm` and `-mlm`.

4.4.1 Exposure/outcome Mode

The first mode uses the `-exposure` parameter to specify the exposure data required by the Mendelian randomization model, `-outcome` provides the outcome data required by the Mendelian randomization model, and `-qtl` provides the genetic variation of the exposure data, then Mendelian randomization analysis can be performed through the below command:

```
MRBIGR.py mr -g geno_qc -exposure exposure.csv -qtl exposure_qtl.csv -outcome outcome.csv -mlm -thread 12 -o mr_out
```

The subcommand `mr` invokes the Mendelian randomization analysis module; parameter `-exposure` is the CSV format exposure data; `-outcome` is the CSV format outcome data; `-qtl` is the CSV format exposure QTL data; `-thread` is the thread number to run the command; `-g` is the plink-bed format input genotype data; `-mlm` represents perform Mendelian randomization analysis through mixed linear model; `-o` is the prefix of output file. After this step, a MR result file named *mr_out.MR.csv* would be generated.

4.4.2 Pairwise Mode

The second mode uses the `-gene_exp` parameter to specify the population gene expression data, `-qtl` specifies the genetic variation that affects gene expression, and `-pairwise` indicates to perform Mendelian randomization analysis between all pairs of genes in the population gene expression data, pairwise genes Mendelian randomization analysis can be performed through the below command:

```
MRBIGR.py mr -g geno_qc -gene_exp gene_exp.csv -pairwise -mlm -qtl gene_qtl.csv -thread 12  -o pairwise_mr_out
```

The subcommand mr invokes the Mendelian randomization analysis module; parameter `-gene_exp` is the CSV format population gene expression data; `-qtl` is the CSV format gene QTL data; `-thread` is the thread number to run the command; `-g` is the plink-bed format input genotype data; `-mlm` represents perform Mendelian randomization analysis through mixed linear model; `-o` is the prefix of output file. After this step, a MR result file named *pairwise_mr_out.MR.csv* would be generated.

4.4.3 TF Mode

The third mode uses the -gene_exp parameter to specify the population expression data while using the -tf and -target parameters to specify the transcription factor and their target genes, respectively, and then perform Mendelian randomization analysis between the transcription factors and the target genes. The regulatory relationship between transcription factors and target genes can be divided into direct and indirect regulatory relationships. In detail, when the genomic distance between a transcription factor and its target gene is < 500 kb, the relationship between them is defined as direct relationship, while the regulatory relationship between the transcription factor and remaining target genes are defined as indirect regulation. The -type parameter is used to specify the regulatory relationship between transcription factors and target genes in Mendelian randomization analysis. The command line is as follows:

```
MRBIGR.py mr -g geno_qc -gene_exp gene_exp.csv -tf tf_genefunc.txt -target target_genefunc.txt -mlm -qtl tf_qtl.csv -type direct -threads 12 -o tf_mr_out
```

The subcommand mr invokes the Mendelian randomization analysis module; parameter -gene_exp is the CSV format population gene expression data; -qtl is the CSV format transcription factor QTL data; -tf is the transcription factor annotation file; -target is the annotation files for genes targeted by transcription factors; -thread is the thread number to run the command; -g is the plink-bed format input genotype data; -mlm represents perform Mendelian randomization analysis through mixed linear model; -type is the regulatory relationship between transcription factors and target genes, with either direct (perform Mendelian randomization analysis transcription factors and directly regulated targeted genes) or all (perform Mendelian randomization analysis between transcription factors and all targeted genes) as option;-o is the prefix of output file. After this step, a MR result file named *tf_mr_out.MR.csv* would be generated.

Tips: the transcription factor annotation file and the annotation files for genes targeted by transcription factors could be annotated use a four columns gene annotation file in TSV format, which looks like this:

```
geneid  aliased position    function
Zm00001eb015280 Zm00001eb015280 1:52319290-52320913:+   Zm00001eb015280
Zm00001eb000610 Zm00001eb000610 1:2555438-2555822:+     Zm00001eb000610
Zm00001eb033210 Zm00001eb033210 1:184900367-184903962:+ Zm00001eb033210
```

4.5 MR-based network construction and module identification

In MRBIGR, the weight of MR effects between gene pairs are used to construct the MR-based network and ClusterONE algorithm is used to identify modules for the constructed network. The MR-based network analysis command line is as follows:

```
MRBIGR.py net -mr pairwise_mr_out.MR.csv -plot -o net_out
```

The subcommand `net` invokes the Mendelian randomization based network analysis module; parameter `-mr` is the CSV format Mendelian randomization analysis data; `-plot` represents plot network for each identified network module ;`-o` is the prefix of output file. After this step, network edgelist file *`net_out.edge_list`*, ClusterONE software result *`net_out.clusterone.result.csv`*, network module plot *`net_out.module*.pdf`* and final network module file *`net_out.module.csv`* would be generated.

4.6 Gene ontology analysis of network modules

Gene ontology analysis of the MR-based network modules is helpful to find modules with biological significance. The *enrichGO* function in R package *clusterProfiler* is used to perform GO enrichment analysis on each module based on gene ontology annotation, and rich graphics (e.g., `dotplot`, `barplot`, `cnetplot`, `heatplot`, `emapplot`, `upsetplot`) could be chose for the visualization of the enrichment results. The command line is as follows:

```
MRBIGR.py go -gene_lists net_out.module.csv -go_anno go_anno.txt -
gene_info gene_anno.txt -plot_type barplot,dotplot -o go_out
```

The subcommand `go` invokes the gene ontology enrichment analysis module; parameter `-gene_lists` is the CSV format network module data; `-go` is the Tabular format gene ontology annotation of each gene data; `-gene_info` is the Tabular format gene annotation data; `-plot_type` is visualization type of enrichment results; `-o` is the prefix of output file. After this step, gene ontology enrichment analysis result *`go_out.GO.csv`*, visualized results of functional enrichment results *`go_out.BP.dotplot.pdf`*, *`go_out.MF.dotplot.pdf`* and *`go_out.CC.dotplot.pdf`*, *`go_out.BP.barplot.pdf`*, *`go_out.MF.barplot.pdf`* and *`go_out.CC.barplot.pdf`* would be generated.

Tips: the gene ontology annotation information of each gene could be annotated use a two columns annotation file in TSV format, which looks like this:

```
Zm00001d018305  GO:0051649
Zm00001d018305  GO:0003677
Zm00001d003209  GO:0005886
Zm00001d003209  GO:0044238
```

The gene annotation information of each gene could by annotated use a five columns annotation file in TSV format, which looks like this:

```
    gene_id chr start    end stard    annotation
    Zm00001d027230  1    44289    49837    +    Mitochondrial transcription termination
factor family protein
    Zm00001d027231  1    50877    55716    -    OSJNBa0093O08.6 protein
```

```
Zm00001d027232  1   92299   95134   -   Zm00001d027232
Zm00001d027234  1   118683  119739  -   Zm00001d027234
```

4.7 Data visualization

The data visualization module in MRBIGR provides a collection of plot functions for the visualization of genotype, phenotype, GWAS, Mendelian randomization, network, and GO analysis results. This module can be invoked through the subcommand plot. The parameter `-i` is used to specify the data file needed for plotting, `-plot_type` specifies the plot type, the options include `manhattan`, `qqplot`, `SNPdensity`, `hist`, `boxplot`, `scatterplot_ps`, `barplot`, `dotplot`, `forestplot`, and `scatterplot_mr`.

4.7.1 Genotype data based plot

The genotype data based plot includes SNP density plot (`SNPdensity`) and scatterplot of population (`scatter_ps`). SNP density plot is used to demonstrate the distribution of the SNP across the genome. The command line is as follows:

```
MRBIGR.py plot -i example.snp_inofo.txt -plot_type SNPdensity -o plot_out
```

Tips: The SNP information file could be annotated use a three columns file in TSV format, wh ich looks like this:

```
rs  chr ps
chr10.s_109149  10  109149
chr10.s_109277  10  109277
chr10.s_109475  10  109475
chr10.s_109511  10  109511
```

The first column represents the SNP id, the second column represents the chromosome of the SNP, and the third column represents the position of the SNP.

The population scatterplot takes a PCA or t-SNE result file as input to visualize the population structure. The population information of each individual can be specified by the -group parameter, which is used to assign individuals to populations with different colors. The command line is as follows:

```
MRBIGR.py plot -i example.PCA.csv -plot_type scatter_ps -
group example.ps_info.csv -o plot_out
```

The population structure file could be annotated use a three columns file in CSV format, which looks like this:

```
RIL,PC1,PC2
GEMS58,38.88 ,-59.69
CML415,-30.07 ,1.74
```

```
SI273,52.52 ,11.30
CIMBL135,-38.45 ,9.53
```

The first column represents the sample id, the second column represents first principal component data of each inbred line, and the third column represents second principal component data of each inbred line.

The group file could be annotated use a two columns file in CSV format, which looks like this:

```
RIL,subpop
GEMS58,NSS
CML415,TST
SI273,NSS
CIMBL135,TST
```

The first column represents the sample identifier, the second column represents the population information of each sample.

4.7.2 Phenotype data based plot

The phenotype data based plot includes histogram (`hist`) and boxplot (`boxplot`), which are used to demonstrate the distribution of phenotype data. Boxplot can not only show the overall distribution of phenotypes, but also the phenotype distribution of different groups, when the -group parameter is specified. The command line is as follows:

```
MRBIGR.py plot -i example.phe.csv -plot_type hist -o plot_out
MRBIGR.py plot -i example.phe.csv -plot_type boxplot -group example.group.csv -o plot_out
```

The phenotype file could be annotated use a two columns file in CSV format, which looks like this:

```
RIL,1st
CAU1,0.8234
CAU2,0.7121
CAU3,0.6665
CAU4,0.8731
```

The first column represents the sample identifier, the second column represents the phenotype data of each sample.

The group file could be annotated use a two columns file in CSV format, which looks like this:

```
RIL,haplotype
CAU1,A
CAU2,A
CAU3,A
CAU4,T
```

The first column represents the sample ID, the second column represents the group of each sample, which can be haplotype type or populations information.

4.7.3 GWAS based plot

The GWAS based plot includes manhattan plot (`manhattan`) and QQ plot (`qqplot`), which is used to demonstrate GWAS result. The command line is as follows:

```
MRBIGR.py plot -i example.gwas.txt -plot_type manhattan -o plot_out
MRBIGR.py plot -i example.gwas.txt -plot_type qqplot -o plot_out
```

Tips: The GWAS result file could be annotated use a four columns file in TSV format, which looks like this:

```
rs	chr	ps	p_wald
1_1922301	1	1922301	9.121183e-03
1_1928050	1	1928050	1.795902e-03
1_2521954	1	2521954	7.200593e-03
1_2522874	1	2522874	6.791745e-03
```

The first column represents the ID of SNPs, the second column represents the chromosome of SNPs, and the third column represents the position of SNPs, the fourth column represents the GWAS P-value of SNPs.

4.7.4 MR data based plot

The MR based plot includes forest plot (`forestplot`) and scatter plot (`scatter_mr`). The forest plot is used to display the effect between each exposure and outcome traits, and can specify the order of the outcome traits in the forest plot through the order file, so as to compare the effects of different exposures on the outcome traits. It can be used to compare the influence of different exposures under a small number of outcome traits. The command line is as follows:

```
MRBIGR.py plot -i mr_out.MR.csv -plot_type forestplot -order order.txt -o plot_out
```

Tips: The MR file is generated by mr subcommand. The order file is a column file that list outcome traits, which looks like this:

```
Ear length
Ear leaf width
Kernel width
Ear diameter
```

The scatter plot is used to display the Mendelian randomized p-value between exposure and outcome, and group the outcome through the group file. It can be used to show the impact of exposure on the outcome traits under a large number of outcome traits. The command line is as follows:

```
MRBIGR.py plot -i mr_out.MR.csv -plot_type scatter_mr -group anno.txt -o plot_out
```

Tips: The group file of outcome traits could be annotated use a two columns annotation file in CSV format, which looks like this:

```
id,group
Zm00001d028748,1
Zm00001d025572,10
Zm00001d005932,2
Zm00001d042777,3
```

The first column represents the outcome trait, and the second column represents the category corresponding to the outcome trait, which can be the chromosome where the gene is located, the type of metabolite, *etc*.

7.4.5 GO based plot

The GO based plot includes bar plot (`barplot`) and dot plot (`dotplot`). It depicts the enrichment scores (e.g. P-values) and gene count or ratio in plot. The command line is as follows:

```
MRBIGR.py plot -i example.GO.csv -plot_type barplot -o plot_out
MRBIGR.py plot -i example.GO.csv -plot_type dotplot -o plot_out
```

Tips: The GO enrichment result file could be annotated use a ten columns file in CSV format, which looks like this:

```
ONTOLOGY,ID,Description,GeneRatio,BgRatio,pvalue,p.adjust,qvalue,geneID,Count
MF,GO:0003729,mRNA binding,5/82,371/37229, 1.393e-3,0.033,0.026,Zm00001d044979/Zm00001d049442/Zm00001d005350,5
```

The first column represents the ontology domains, including BP, MF and CC, the second column represents the ontology id, the third column represents the description of ontology id, the fourth column represents the ratio of genes containing ontology id in the gene list to the total genes in the gene list, the fifth column represents the ratio of genes containing ontology id in whole genome to the total genes in whole genome, the sixth column represents the P value of GO enrichment analysis, the seventh and eighth columns represents the corrected P value of GO enrichment analysis, it can be same when the GO enrichment result file is user-customized, the ninth column represents the gene id of genes containing ontology id, the tenth represents the count of genes containing ontology id.