GenEpi Documentation

Chester (Yu-Chuan Chang)

Aug 04, 2020

Contents

GenEpi[1] is a package to uncover epistasis associated with phenotypes by a machine learning approach, developed by Yu-Chuan Chang at c4Lab of National Taiwan University and AILabs.

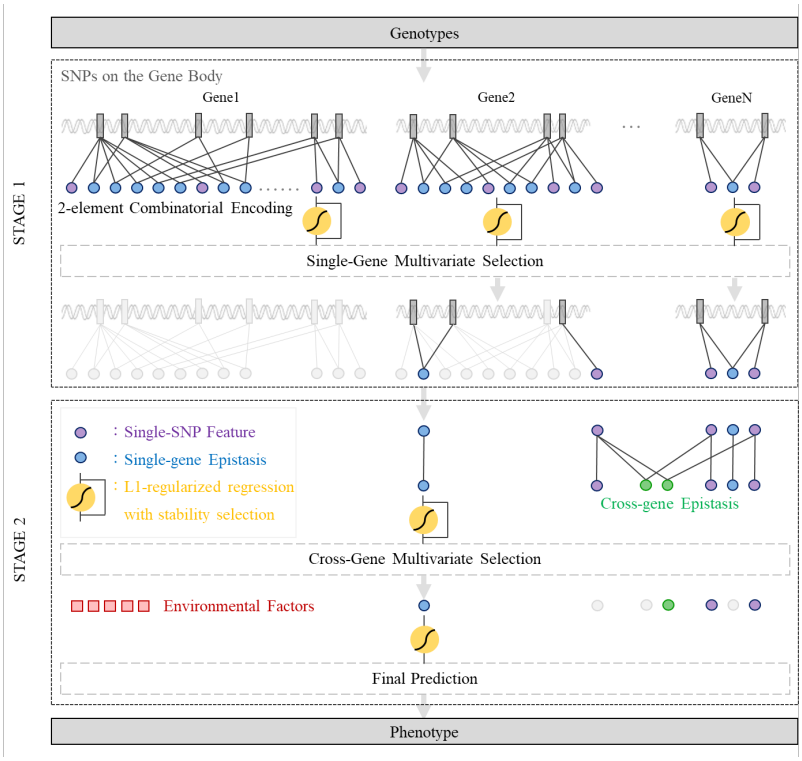

Fig. 1: The architecture and modules of GenEpi.

[1] Yu-Chuan Chang, June-Tai Wu, Ming-Yi Hong, Yi-An Tung, Ping-Han Hsieh, Sook Wah Yee, Kathleen M. Giacomini, Yen-Jen Oyang, and Chien-Yu Chen. "Genepi: Gene-Based Epistasis Discovery Using Machine Learning." BMC Bioinformatics 21, 68 (2020). https://doi.org/10.1186/s12859-020-3368-2

CHAPTER 1

Introduction

GenEpi is designed to group SNPs by a set of loci in the gnome. For examples, a locus could be a gene. In other words, we use gene boundaries to group SNPs. A locus can be generalized to any particular regions in the genome, e.g. promoters, enhancers, etc. GenEpi first considers the genetic variants within a particular region as features in the first stage, because it is believed that SNPs within a functional region might have a higher chance to interact with each other and to influence molecular functions.

GenEpi adopts two-element combinatorial encoding when producing features and models them by L1-regularized regression with stability selection In the first stage (STAGE 1) of GenEpi, the genotype features from each single gene will be combinatorically encoded and modeled independently by L1-regularized regression with stability selection. In this way, we can estimate the prediction performance of each gene and detect within-gene epistasis with a low false positive rate. In the second stage (STAGE 2), both of the individual SNP and the within-gene epistasis features selected by STAGE 1 are pooled together to generate cross-gene epistasis features, and modeled again by L1-regularized regression with stability selection as STAGE 1. Finally, the user can combine the selected genetic features with environmental factors such as clinical features to build the final prediction models.

CHAPTER 2

Citing

Please considering cite the following paper if you use GenEpi in a scientific publication.

2.1 Installation

2.1.1 How to Install

GenEpi supports Python 3.7 and up. Use pip to install GenEpi and its dependencies.

```
$ pip install GenEpi
```

Check that you installed the GenEpi sucessfully.

```
$ GenEpi --help
```

After executed previous command on console, you will see:

```
usage: GenEpi [-h] -g G -p P [-s S] [-o O] [-m {c,r}] [-k K] [-t T]
              [--updatedb] [-b {hg19,hg38}] [--compressld] [-d D] [-r R]

optional arguments:
  -h, --help      show this help message and exit
  -g G            filename of the input .gen file
  -p P            filename of the input phenotype
  -s S            self-defined genome regions
  -o O            output file path
  -m {c,r}        choose model type: c for classification; r for regression
  -k K            k of k-fold cross validation
  -t T            number of threads

update UCSC database:
  --updatedb      enable this function
```

(continues on next page)

(continued from previous page)

```
  -b {hg19,hg38}  human genome build

compress data by LD block:
  --compressld    enable this function
  -d D            threshold for compression: D prime
  -r R            threshold for compression: R square
```

2.1.2 Dependencies

Here is the dependency list for running GenEpi. pip takes care of these dependencies automatically when you install GenEpi.

- numpy >= 1.13.0
- psutil >= 4.3.0
- pymysql >= 0.8.0
- scipy >= 0.19.0
- scikit-learn == 0.21.2

2.1.3 System Requirements

For running a quick test, you could install GenEpi on any laptop e.g. a MacBook. When applying GenEpi on a real whole genome-wide dataset, here are recommended system requirements:

Processor 2.3 GHz Intel XEON® E5-2673 v4 * 32

RAM 256 GiB

Storage 500 GiB

These requirements are refer to the specification of Microsoft Azure E32 v3.

Note: GenEpi is a memory-consuming package, which might cause memory errors when calculating the epistasis of a gene containing a large number of SNPs. We recommend that the memory for running GenEpi should be over 256 GB.

2.2 Quickstart

This section gets you started quickly, the I/O described in I/O File Fomats, more usage examples please find in More Usage Examples, discussing each of the sub-modules introduced in How it Work.

2.2.1 Running a Quick Test

Please use following command to run a quick test, you will obtain all the outputs of GenEpi in your current folder.

```
$ GenEpi -g example -p example -o ./
```

The progress will print on console:

```
step1: Down load UCSC Database. DONE!
step2: Estimate LD. DONE!
Warning of step3: .gen file should be sorted by chromosome and position
step3: Split by gene. DONE!
step4: Detect single gene epistasis. DONE!
step5: Detect cross gene epistasis. DONE! (Training score:0.63; 2-fold Test Score:0.
↪61)
step6: Ensemble with covariates. DONE! (Training score:0.63; 2-fold Test Score:0.60)
```

GenEpi will automatically generate three folders (snpSubsets, singleGeneResult, crossGeneResult) in output path (arg: -o). The following tree structure is the contents of the output folder. You could go to the folder **crossGeneResult** directly to obtain your main result table for episatasis in **Result.csv**.

```
./
├── GenEpi_Log_DATE-TIME.txt
├── crossGeneResult
│   ├── Classifier.pkl
│   ├── Classifier_Covariates.pkl
│   ├── Feature.csv
│   └── Result.csv
├── sample.LDBlock
├── sample.csv
├── sample.gen
├── sample_LDReduced.gen
├── singleGeneResult
│   ├── All_Logistic_k2.csv
│   ├── APOC1_Feature.csv
│   ├── APOC1_Result.csv
│   ├── APOE_Feature.csv
│   ├── APOE_Result.csv
│   ├── PVRL2_Feature.csv
│   ├── PVRL2_Result.csv
│   ├── TOMM40_Feature.csv
│   └── TOMM40_Result.csv
└── snpSubsets
    ├── APOC1_23.gen
    ├── APOE_11.gen
    ├── PVRL2_48.gen
    └── TOMM40_67.gen
```

2.2.2 Interpreting the Main Result Table

Here is the contents of Result.csv, which mean the episatasis seleted by GenEpi.

RSID	Weight	-Log10(χ2 p-value)	Odds Ratio	Genotype Frequency	Gene Symbol
rs157580_BB rs2238681_AA	0.9729	8.4002	9.3952	0.1044	TOMM40
rs449647_AA rs769449_AB	0.7065	8.0278	5.0877	0.2692	APOE
rs59007384_BB rs11668327_AA	1.0807	8.0158	12.0408	0.0824	TOMM40
rs283811_BB rs7254892_AA	1.0807	8.0158	12.0408	0.0824	PVRL2
rs429358_AA	-0.7587	5.7628	0.1743	0.5962	APOE
rs73052335_AA rs429358_AA	-0.7289	5.6548	0.1867	0.5714	APOC1*APOE

We listed the statistical significance of the selected genetic features in Result.csv. The first column lists each feature by its RSID and the genotype (denoted as RSID_genotype), the pairwise epistatis features are represented using two SNPs. The weights in the second column were extracted from the L1-regularized regression model. The last column describes the genes where the SNPs are located according to the genomic coordinates. We used a star sign to denote the epistasis between genes. The p-values of the χ2 test (the quantitative task will use student t-test) are also included. The odds ratio significantly away from 1 also indicates whether the features are potential causal or protective genotypes. Since low genotype frequency may cause unreliable odds ratios, we also listed this information in the table.

2.3 I/O File Fomats

We provided test data sample.gen and sample.csv in example folder. After running a quick test, GenEpi will automatically copy these test data to output path and generate all of the output folders and files as following tree structure. Please see the following detail about the format of these input and output data.

```
./
├── GenEpi_Log_DATE-TIME.txt
├── crossGeneResult
│   ├── Classifier.pkl
│   ├── Classifier_Covariates.pkl
│   ├── Feature.csv
│   └── Result.csv
├── sample.LDBlock
├── sample.csv
├── sample.gen
├── sample_LDReduced.gen
├── singleGeneResult
│   ├── All_Logistic_k2.csv
│   ├── APOC1_Feature.csv
│   ├── APOC1_Result.csv
│   ├── APOE_Feature.csv
│   ├── APOE_Result.csv
│   ├── PVRL2_Feature.csv
│   ├── PVRL2_Result.csv
│   ├── TOMM40_Feature.csv
│   └── TOMM40_Result.csv
└── snpSubsets
    ├── APOC1_23.gen
    ├── APOE_11.gen
```

(continues on next page)

(continued from previous page)

```
    ├── PVRL2_48.gen
    └── TOMM40_67.gen
```

2.3.1 Input: Genotype Data

Sample.gen is an example of genotype data. GenEpi takes the Genotype File Format (.GEN) used by Oxford statistical genetics tools, such as IMPUTE2 and SNPTEST as the input format for genotype data. If your files are in PLINK format (.BED/.BIM/.FAM) or 1000 Genomes Project text Variant Call Format (.VCF), you could use PLINK with the following command to convert them to the .GEN file.

If your files are in the .BED/.BIM/.FAM format.

```
$ plink --bfile prefixOfTheFilename --recode oxford --out prefixOfTheFilename
```

If your file is in the .VCF format.

```
$ plink --vcf filename.vcf --recode oxford --out prefixOfTheFilename
```

2.3.2 Input: Phenotype Data

Sample.gen is an example of phenotype data with environmental factor (e.g. the age of each sample). GenEpi takes the common .CSV file without header line as the input format for phenotype and environmental factor data. The last column of the file will be considered as the phenotype data (e.g. 1 or 0, which indicate case/control, respectively) and the other columns will be considered as the environmental factor data.

Warning: The sequential order of the phenotype data should be the same as that in the .GEN file.

2.3.3 Output: LDBlock File

GenEpi has a moudle, which can reduce the dimension of the input feature by estimating the linkage disequilibrium (LD). After performing dimension reduction, GenEpi will generate two files, a dimension-reduced .GEN file and a file containing LD blocks (.LDBlock file). Each row in the .LDBlock file indicates a LD block (see below for examples). The SNPs in front of colon signs are the representative SNPs of each LD block, and only these SNPs will be retained in the dimension-reduced .GEN file.

```
rs429358:rs429358
rs7412:rs7412
rs117656888:rs117656888
rs1081105:rs1081105
rs1081106:rs1081106,rs191315680
```

2.3.4 Output: SnpSubsets Folder

Since GenEpi is a gene-based epistasis discovering method, the input genotype will first be splited into group of each gene. The subsets of the .GEN file for each gene will be stored in the folder snpSubsets. The naming rule of the filename is GeneSymbol_NumberOfVariantsOnGene.gen

2.3.5 Output: SingleGeneResult Folder

In first stage of GenEpi, all the .GEN file for each gene will be modeled gene by gene. Every models will output two kinds of data the GeneSymbol_Result.csv and GeneSymbol_Feature.csv. The format of GeneSymbol_Result.csv please refer to the section Interpreting the Main Result Table. The only difference is the GeneSymbol_Result.csv is a result table for single gene. Moreover, GeneSymbol_Feature.csv are the raw features corresponds to the episatsis in GeneSymbol_Result.csv. Beside these two types of files, there is a file named All_Logistic/Lasso.csv, which contains all the prediction scores of each gene, please see as following example.

```
$ head All_Logistic_k2.csv

GeneSymbol,F1Score
PVRL2,0.5745454545454547
APOC1,0.5681818181818181
TOMM40,0.602510460251046
APOE,0.592
```

2.3.6 Output: CrossGeneResult Folder

The results of the second stage of GenEpi - cross gene modeling will be generated in this folder. The formats are as same as the description in previous section *SingleGeneResult Folder*. Moreover, the final models will be persisted in this folder as Classifier/Regressor.pkl and Classifier/Regressor_Covariates.pkl, respectively. You could keep these models for future use without reconstructing them.

2.3.7 Output: Porcess Log

The performance of genetic feature only and genetic + environmental factor models will be logged into GenEpi_Log_DATE-TIME.txt. Other process information such as the setting of arguments, the time cost will also be recorded.

```
start analysis at: 20191009-17:54:45

Arguments in effect:
        -g (input genotype filename): ./sample.gen
        -p (input phenotype filename): ./sample.csv
        -s (self-defined genome regions): None
        -o (output filepath): ./

        -m (model type): Classification -k (k-fold cross validation): 2
        -t (number of threads): 4

        --updatedb (enable function of update UCSC database): True
        -b (human genome build): hg19

        --compressld (enable function of LD data compression): True
        -d (D prime threshold): 0.9
        -r (R square threshold): 0.9

Number of variants: 223
Number of samples: 364
Overall genetic feature performance (F1 score)
Training: 0.6307053941908715
Testing (2-fold CV): 0.6134453781512604
```

(continues on next page)

(continued from previous page)

```
Ensemble with co-variate performance (F1 score)
Training: 0.632
Testing (2-fold CV): 0.6016260162601627

end analysis at: 20191009-17:54:58
```

2.3.8 Seld-defined Genome Regions

GenEpi supports seld-defined genome regions for first stage to subset the data. Please prepare your genome regions in .TXT with the columns [chromosome, start, end, strand, geneSymbol], for eample:

```
1,10873,14409,+,DDX11L1
1,14361,30370,-,WASH7P
1,34610,37081,-,FAM138F
1,68090,70008,+,OR4F5
...
```

2.4 More Usage Examples

For checking all the optional arguments, please use –help.

```
$ GenEpi --help
```

You will obtain the following argument list:

```
usage: GenEpi [-h] -g G -p P [-s S] [-o O] [-m {c,r}] [-k K] [-t T]
              [--updatedb] [-b {hg19,hg38}] [--compressld] [-d D] [-r R]

optional arguments:
  -h, --help      show this help message and exit
  -g G            filename of the input .gen file
  -p P            filename of the input phenotype
  -s S            self-defined genome regions
  -o O            output file path
  -m {c,r}        choose model type: c for classification; r for regression
  -k K            k of k-fold cross validation
  -t T            number of threads

update UCSC database:
  --updatedb      enable this function
  -b {hg19,hg38}  human genome build

compress data by LD block:
  --compressld    enable this function
  -d D            threshold for compression: D prime
  -r R            threshold for compression: R square
```

2.4.1 Applying on Your Data

```
$ GenEpi -g full_path_of_your_.GEN_file -p full_path_of_your_.CSV_file -o ./
```

2.4.2 For Quantitative Study

GenEpi can support both case/control and quantitative studies, for quantitative studies please modify the parameter -m. You could download the test data and excute the following command.

```
$ GenEpi -g sample.gen -p sample_q.csv -o ./ -m r
```

2.4.3 Changing the Genome Build

For changing the build of USCS genome browser, please modify the parameter -b.

```
$ GenEpi -g example -p example -o ./ --updatedb -b hg38
```

2.4.4 Threshold for Dimension Reduction

You could modify the threshold for Linkage Disequilibrium dimension reduction by following command.

```
$ GenEpi -g example -p example -o ./ --compressld -d 0.9 -r 0.9
```

2.4.5 Self-defined Genome Region

Please prepare your self-defined genome region in this format. Then, use the parameter -s for applying it on your data.

```
$ GenEpi -s full_path_of_your_genome_region_file -g full_path_of_your_.GEN_file -p␣
↪full_path_of_your_.CSV_file -o ./
```

2.5 How it Works

The main procedures of GenEpi described in Introduction. In addition to the main procedures, two pre-processing steps are also implemented in GenEpi: retrieving the gene information from public databases and reducing the gene information from public databases and reducing the dimensionality of the features using linkage disequilibrium. All of the precedures be implemented as six steps in GenEpi submoudles. The interations bewteen the I/O and these submodules please find in the figure below. For API documentations of these submodules please find in next section.

2.6 API Documentations

2.6.1 GenEpi module

Created on Apr 2019

@author: Chester (Yu-Chuan Chang)

genepi.GenEpi.**ArgumentsParser**()
To obtain and parse the arguments from user.

Parameters **None** –

Returns argparse.ArgumentParser

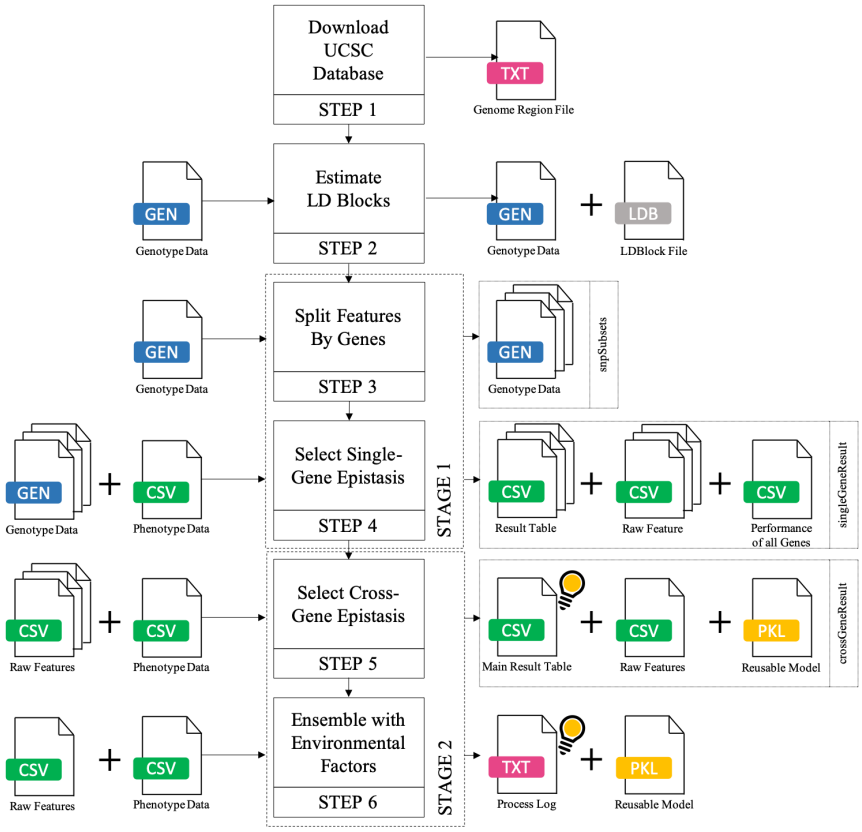

Fig. 1: The interations bewteen the I/O and GenEpi submodules.

genepi.GenEpi.**InputChecking**(*str_inputFileName_genotype*, *str_inputFileName_phenotype*, *args*)
To check the numbers of sample are consistent in genotype and phenotype data.

Parameters

- **str_inputFileName_genotype** (*str*) – File name of input genotype data
- **str_inputFileName_phenotype** (*str*) – File name of input phenotype data

Returns

tuple containing:

- int_num_genotype (int): The sample number of genotype data
- int_num_phenotype (int): The sample number of phenotype data

Return type (tuple)

genepi.GenEpi.**main**(*args=None*)
Main function for obtaining user arguments, controling workflow and recording log file.

Parameters **None** –

Returns None

2.6.2 step1_downloadUCSCDB

Created on Feb 2018

@author: Chester (Yu-Chuan Chang)

genepi.step1_downloadUCSCDB.**DownloadUCSCDB**(*str_outputFilePath='/home/docs/checkouts/readthedocs.org/user_build packages/genepi-2.0.10-py3.7.egg/genepi'*, *str_hgbuild='hg19'*)
To obtain the gene information such as official gene symbols and genomic coordinates, this function is for retrieving kgXref and knownGene data table from the UCSC human genome annotation database

Parameters

- **str_outputFilePath** (*str*) – File path of output database
- **str_hgbuild** (*str*) – Genome build (eg. "hg19")

Returns

- Expected Success Response:

```
"step1: Down load UCSC Database. DONE!"
```

2.6.3 step2_estimateLD

Created on Feb 2018

@author: Chester (Yu-Chuan Chang)

genepi.step2_estimateLD.**EstimateAlleleFrequency**(*gen_snp*)
A function for estimating allele frequency of a single varaint

Parameters **gen_snp** (*list*) – The genotypes of a variant of all samples

Returns

tuple containing:

- float_frequency_A (float): The reference allele type frequency
- float_frequency_B (float): The alternative allele type frequency

Return type (tuple)

genepi.step2_estimateLD.**EstimateLDBlock**(*str_inputFileName_genotype*, *str_outputFilePath=''*, *float_threshold_DPrime=0.8*, *float_threshold_RSquare=0.8*)

A function for implementing linkage disequilibrium (LD) dimension reduction. In genotype data, a variant often exhibits high dependency with its nearby variants because of LD. In the practical implantation, we prefer to group these dependent features to reduce the dimension of features. In other words, we can take the advantages of LD to reduce the dimensionality of genetic features. In this regard, this function adopted the same approach developed by Lewontin (1964) to estimate LD. We used D' and r2 as the criteria to group highly dependent genetic features as blocks. In each block, we chose the features with the largest minor allele frequency to represent other features in the same block.

Parameters

- **str_inputFileName_genotype** (*str*) – File name of input genotype data
- **str_outputFilePath** (*str*) – File path of output file
- **float_threshold_DPrime** (*float*) – The Dprime threshold for discriminating a LD block (default: 0.8)
- **float_threshold_RSquare** (*float*) – The RSquare threshold for discriminating a LD block (default: 0.8)

Returns

- Expected Success Response:

```
"step2: Estimate LD. DONE!"
```

genepi.step2_estimateLD.**EstimatePairwiseLD**(*gen_snp_1*, *gen_snp_2*)

Lewontin (1964) linkage disequilibrium (LD) estimation.

Parameters

- **gen_snp_1** (*list*) – The genotypes of first variant of all samples
- **gen_snp_2** (*list*) – The genotypes of second variant of all samples

Returns

tuple containing:

- float_D_prime (float): The DPrime of these two variants
- float_R_square (float): The RSquare of these two variants

Return type (tuple)

2.6.4 step3_splitByGene

Created on Feb 2018

@author: Chester (Yu-Chuan Chang)

genepi.step3_splitByGene.**SplitByGene**(*str_inputFileName_genotype*, *str_inputFileName_UCSCDB='/home/docs/checkouts/readthedocs.org/user_bu packages/genepi-2.0.10-py3.7.egg/genepi/UCSCGenomeDatabase.txt'*, *str_outputFilePath=''*)

In order to extract genetic features for a gene, this function used the start and end positions of each gene from the local UCSC database to split the genetic features. Then, generate the .GEN files for each gene in the folder named snpSubsets.

Parameters

- **str_inputFileName_genotype** (*str*) – File name of input genotype data
- **str_inputFileName_UCSCDB** (*str*) – File name of input genome regions
- **str_outputFilePath** (*str*) – File path of output file

Returns

- Expected Success Response:

```
"step3: Split by gene. DONE!"
```

Warning: "Warning of step3: .gen file should be sorted by chromosome and position"

genepi.step3_splitByGene.**SplitMegaGene**(*list_snpsOnGene*, *int_window*, *int_step*, *str_outputFilePath*, *str_outputFileName*)

In order to extract genetic features for a gene, this function used the start and end positions of each gene from the local UCSC database to split the genetic features. Then, generate the .GEN files for each gene in the folder named snpSubsets.

Parameters

- **list_snpsOnGene** (*list*) – A list contains SNPs on a gene
- **int_window** (*int*) – The size of the sliding window
- **int_step** (*int*) – The step of the sliding window
- **str_outputFilePath** (*str*) – File path of output file
- **str_outputFileName** (*str*) – File name of output file

Returns None

2.6.5 step4_singleGeneEpistasis_Lasso

Created on Feb 2018

@author: Chester (Yu-Chuan Chang)

genepi.step4_singleGeneEpistasis_Lasso.**BatchSingleGeneEpistasisLasso**(*str_inputFilePath_genotype*, *str_inputFileName_phenotype*, *str_outputFilePath=''*, *int_kOfKFold=2*, *int_nJobs=2*)

Batch running for the single gene workflow.

Parameters

- **str_inputFilePath_genotype** (*str*) – File path of input genotype data

- **str_inputFileName_phenotype** (*str*) – File name of input phenotype data
- **str_outputFilePath** (*str*) – File path of output file
- **int_kOfKFold** (*int*) – The k for k-fold cross validation (default: 2)
- **int_nJobs** (*int*) – The number of thread (default: 1)

Returns

- Expected Success Response:

```
"step4: Detect single gene epistasis. DONE!"
```

genepi.step4_singleGeneEpistasis_Lasso.**FeatureEncoderLasso**(*np_genotype_rsid*, *np_genotype*, *np_phenotype*, *int_dim*)

Implementation of the two-element combinatorial encoding.

Parameters

- **np_genotype_rsid** (*ndarray*) – 1D array containing rsid of genotype data with *str* type
- **np_genotype** (*ndarray*) – 2D array containing genotype data with *int8* type
- **np_phenotype** (*ndarray*) – 2D array containing phenotype data with *float* type
- **int_dim** (*int*) – The dimension of a variant (default: 3. AA, AB and BB)

Returns

tuple containing:

- list_interaction_rsid (ndarray): 1D array containing rsid of genotype data with *str* type
- np_interaction (ndarray): 2D array containing genotype data with *int8* type

Return type (tuple)

genepi.step4_singleGeneEpistasis_Lasso.**FilterInLoading**(*np_genotype*, *np_phenotype*)

This function is for filtering low quality varaint. Before modeling each subset of genotype features, two criteria were adopted to exclude low quality data. The first criterion is that the genotype frequency of a feature should exceed 5%, where the genotype frequency means the proportion of genotype among the total samples in the dataset. The second criterion is regarding the association between the feature and the phenotype. We used $\chi 2$ test to estimate the association between the feature and the phenotype, and the p-value should be smaller than 0.01.

Parameters

- **np_genotype** (*ndarray*) – 2D array containing genotype data with *int8* type
- **np_phenotype** (*ndarray*) – 2D array containing phenotype data with *float* type

Returns

np_genotype

2D array containing genotype data with *int8* type

Return type (ndarray)

genepi.step4_singleGeneEpistasis_Lasso.**LassoRegressionCV**(*np_X*, *np_y*, *int_kOfKFold=2*, *int_nJobs=1*)

Implementation of the L1-regularized Lasso regression with k-fold cross validation.

Parameters

- **np_X** (*ndarray*) – 2D array containing genotype data with *int8* type
- **np_y** (*ndarray*) – 2D array containing phenotype data with *float* type
- **int_kOfKFold** (*int*) – The k for k-fold cross validation (default: 2)
- **int_nJobs** (*int*) – The number of thread (default: 1)

Returns

estimator.scores

1D array containing the scores of each genetic features with *float* type

Return type (ndarray)

genepi.step4_singleGeneEpistasis_Lasso.**RandomizedLassoRegression**(*np_X*, *np_y*)

Implementation of the stability selection.

Parameters

- **np_X** (*ndarray*) – 2D array containing genotype data with *int8* type
- **np_y** (*ndarray*) – 2D array containing phenotype data with *float* type

Returns

estimator.scores

1D array containing the scores of each genetic features with *float* type

Return type (ndarray)

genepi.step4_singleGeneEpistasis_Lasso.**SingleGeneEpistasisLasso**(*str_inputFileName_genotype*, *str_inputFileName_phenotype*, *str_outputFilePath=''*, *int_kOfKFold=2*, *int_nJobs=1*)

A workflow to model a single gene containing two-element combinatorial encoding, stability selection, filtering low quality varaint and L1-regularized Lasso regression with k-fold cross validation.

Parameters

- **str_inputFileName_genotype** (*str*) – File name of input genotype data
- **str_inputFileName_phenotype** (*str*) – File name of input phenotype data
- **str_outputFilePath** (*str*) – File path of output file
- **int_kOfKFold** (*int*) – The k for k-fold cross validation (default: 2)
- **int_nJobs** (*int*) – The number of thread (default: 1)

Returns

float_AVG_S_P

The average of the Peason's and Spearman's correlation of the model

Return type (float)

2.6.6 step4_singleGeneEpistasis_Logistic

Created on Feb 2018

@author: Chester (Yu-Chuan Chang)

genepi.step4_singleGeneEpistasis_Logistic.**BatchSingleGeneEpistasisLogistic**(*str_inputFilePath_geno*, *str_inputFileName_phe*, *str_outputFilePath=''*, *int_kOfKFold=2*, *int_nJobs=2*)

Batch running for the single gene workflow.

Parameters

- **str_inputFilePath_genotype** (*str*) – File path of input genotype data
- **str_inputFileName_phenotype** (*str*) – File name of input phenotype data
- **str_outputFilePath** (*str*) – File path of output file
- **int_kOfKFold** (*int*) – The k for k-fold cross validation (default: 2)
- **int_nJobs** (*int*) – The number of thread (default: 1)

Returns

- Expected Success Response:

```
"step4: Detect single gene epistasis. DONE!"
```

genepi.step4_singleGeneEpistasis_Logistic.**FeatureEncoderLogistic**(*np_genotype_rsid*, *np_genotype*, *np_phenotype*, *int_dim*)

Implementation of the two-element combinatorial encoding.

Parameters

- **np_genotype_rsid** (*ndarray*) – 1D array containing rsid of genotype data with *str* type
- **np_genotype** (*ndarray*) – 2D array containing genotype data with *int8* type
- **np_phenotype** (*ndarray*) – 2D array containing phenotype data with *float* type
- **int_dim** (*int*) – The dimension of a variant (default: 3. AA, AB and BB)

Returns

tuple containing:

- list_interaction_rsid (ndarray): 1D array containing rsid of genotype data with *str* type
- np_interaction (ndarray): 2D array containing genotype data with *int8* type

Return type (tuple)

genepi.step4_singleGeneEpistasis_Logistic.**FilterInLoading**(*np_genotype*, *np_phenotype*)

This function is for filtering low quality varaint. Before modeling each subset of genotype features, two criteria were adopted to exclude low quality data. The first criterion is that the genotype frequency of a feature should exceed 5%, where the genotype frequency means the proportion of genotype among the total samples in the dataset. The second criterion is regarding the association between the feature and the phenotype. We used $\chi 2$

test to estimate the association between the feature and the phenotype, and the p-value should be smaller than 0.01.

Parameters

- **np_genotype** (*ndarray*) – 2D array containing genotype data with *int8* type
- **np_phenotype** (*ndarray*) – 2D array containing phenotype data with *float* type

Returns

np_genotype

2D array containing genotype data with *int8* type

Return type (ndarray)

genepi.step4_singleGeneEpistasis_Logistic.**GenerateContingencyTable**(*np_genotype*, *np_phenotype*)

Generating the contingency table for chi-square test.

Parameters

- **np_X** (*ndarray*) – 2D array containing genotype data with *int8* type
- **np_y** (*ndarray*) – 2D array containing phenotype data with *float* type

Returns

np_contingency

2D array containing the contingency table with *int* type

Return type (ndarray)

genepi.step4_singleGeneEpistasis_Logistic.**LogisticRegressionL1CV**(*np_X*, *np_y*, *int_kOfKFold=2*, *int_nJobs=1*)

Implementation of the L1-regularized Logistic regression with k-fold cross validation.

Parameters

- **np_X** (*ndarray*) – 2D array containing genotype data with *int8* type
- **np_y** (*ndarray*) – 2D array containing phenotype data with *float* type
- **int_kOfKFold** (*int*) – The k for k-fold cross validation (default: 2)
- **int_nJobs** (*int*) – The number of thread (default: 1)

Returns

estimator.scores

1D array containing the scores of each genetic features with *float* type

Return type (ndarray)

genepi.step4_singleGeneEpistasis_Logistic.**RandomizedLogisticRegression**(*np_X*, *np_y*)

Implementation of the stability selection.

Parameters

- **np_X** (*ndarray*) – 2D array containing genotype data with *int8* type
- **np_y** (*ndarray*) – 2D array containing phenotype data with *float* type

Returns

estimator.scores

1D array containing the scores of each genetic features with *float* type

Return type (ndarray)

genepi.step4_singleGeneEpistasis_Logistic.**SingleGeneEpistasisLogistic**(*str_inputFileName_genotype*, *str_inputFileName_phenotype*, *str_outputFilePath=''*, *int_kOfKFold=2*, *int_nJobs=1*)

A workflow to model a single gene containing two-element combinatorial encoding, stability selection, filtering low quality varaint and L1-regularized Logistic regression with k-fold cross validation.

Parameters

- **str_inputFileName_genotype** (*str*) – File name of input genotype data
- **str_inputFileName_phenotype** (*str*) – File name of input phenotype data
- **str_outputFilePath** (*str*) – File path of output file
- **int_kOfKFold** (*int*) – The k for k-fold cross validation (default: 2)
- **int_nJobs** (*int*) – The number of thread (default: 1)

Returns

float_f1Score

The F1 score of the model

Return type (float)

2.6.7 step5_crossGeneEpistasis_Lasso

Created on Feb 2018

@author: Chester (Yu-Chuan Chang)

genepi.step5_crossGeneEpistasis_Lasso.**CrossGeneEpistasisLasso**(*str_inputFilePath_feature*, *str_inputFileName_phenotype*, *str_inputFileName_score=''*, *str_outputFilePath=''*, *int_kOfKFold=2*, *int_nJobs=1*)

A workflow to model a cross gene epistasis containing two-element combinatorial encoding, stability selection, filtering low quality varaint and L1-regularized Lasso regression with k-fold cross validation.

Parameters

- **str_inputFilePath_feature** (*str*) – File path of input feature files from stage 1 - singleGeneEpistasis
- **str_inputFileName_phenotype** (*str*) – File name of input phenotype data
- **str_inputFileName_score** (*str*) – File name of input score file from stage 1 - singleGeneEpistasis
- **str_outputFilePath** (*str*) – File path of output file
- **int_kOfKFold** (*int*) – The k for k-fold cross validation (default: 2)

- **int_nJobs** (*int*) – The number of thread (default: 1)

Returns

tuple containing:

- float_AVG_S_P_train (float): The average of the Peason's and Spearman's correlation of the model for training set
- float_AVG_S_P_test (float): The average of the Peason's and Spearman's correlation of the model for testing set
- Expected Success Response:

```
"step5: Detect cross gene epistasis. DONE!"
```

Return type (tuple)

genepi.step5_crossGeneEpistasis_Lasso.**LassoRegression**(*np_X*, *np_y*, *int_nJobs=1*)

Implementation of the L1-regularized Lasso regression with k-fold cross validation.

Parameters

- **np_X** (*ndarray*) – 2D array containing genotype data with *int8* type
- **np_y** (*ndarray*) – 2D array containing phenotype data with *float* type
- **int_nJobs** (*int*) – The number of thread (default: 1)

Returns

float_AVG_S_P

The average of the Peason's and Spearman's correlation of the model

Return type (float)

genepi.step5_crossGeneEpistasis_Lasso.**RegressorModelPersistence**(*np_X*, *np_y*, *str_outputFilePath=''*, *int_nJobs=1*)

Dumping regressor for model persistence

Parameters

- **np_X** (*ndarray*) – 2D array containing genotype data with *int8* type
- **np_y** (*ndarray*) – 2D array containing phenotype data with *float* type
- **str_outputFilePath** (*str*) – File path of output file
- **int_nJobs** (*int*) – The number of thread (default: 1)

Returns None

2.6.8 step5_crossGeneEpistasis_Logistic

Created on Feb 2018

@author: Chester (Yu-Chuan Chang)

genepi.step5_crossGeneEpistasis_Logistic.**ClassifierModelPersistence**(*np_X*, *np_y*, *str_outputFilePath=''*, *int_nJobs=1*)

Dumping classifier for model persistence

Parameters

- **np_X** (*ndarray*) – 2D array containing genotype data with *int8* type
- **np_y** (*ndarray*) – 2D array containing phenotype data with *float* type
- **str_outputFilePath** (*str*) – File path of output file
- **int_nJobs** (*int*) – The number of thread (default: 1)

Returns None

genepi.step5_crossGeneEpistasis_Logistic.**CrossGeneEpistasisLogistic**(*str_inputFilePath_feature*, *str_inputFileName_phenotype*, *str_inputFileName_score=''*, *str_outputFilePath=''*, *int_kOfKFold=2*, *int_nJobs=1*)

A workflow to model a cross gene epistasis containing two-element combinatorial encoding, stability selection, filtering low quality varaint and L1-regularized Logistic regression with k-fold cross validation.

Parameters

- **str_inputFilePath_feature** (*str*) – File path of input feature files from stage 1 - singleGeneEpistasis
- **str_inputFileName_phenotype** (*str*) – File name of input phenotype data
- **str_inputFileName_score** (*str*) – File name of input score file from stage 1 - singleGeneEpistasis
- **str_outputFilePath** (*str*) – File path of output file
- **int_kOfKFold** (*int*) – The k for k-fold cross validation (default: 2)
- **int_nJobs** (*int*) – The number of thread (default: 1)

Returns

tuple containing:

- float_f1Score_train (float): The F1 score of the model for training set
- float_f1Score_test (float): The F1 score of the model for testing set
- Expected Success Response:

```
"step5: Detect cross gene epistasis. DONE!"
```

Return type (tuple)

genepi.step5_crossGeneEpistasis_Logistic.**GenerateContingencyTable**(*np_genotype*, *np_phenotype*)

Generating the contingency table for chi-square test.

Parameters

- **np_X** (*ndarray*) – 2D array containing genotype data with *int8* type

- **np_y** (*ndarray*) – 2D array containing phenotype data with *float* type

Returns

np_contingency

2D array containing the contingency table with *int* type

Return type (ndarray)

genepi.step5_crossGeneEpistasis_Logistic.**LogisticRegressionL1**(*np_X*, *np_y*, *int_nJobs=1*)

Implementation of the L1-regularized Logistic regression with k-fold cross validation.

Parameters

- **np_X** (*ndarray*) – 2D array containing genotype data with *int8* type
- **np_y** (*ndarray*) – 2D array containing phenotype data with *float* type
- **int_nJobs** (*int*) – The number of thread (default: 1)

Returns

float_f1Score

The F1 score of the model

Return type (float)

genepi.step5_crossGeneEpistasis_Logistic.**PlotPolygenicScore**(*np_X*, *np_y*, *int_kOfKFold=2*, *int_nJobs=1*, *str_outputFilePath=''*)

Plot figure for polygenic score, including group distribution and prevalence to PGS

Parameters

- **np_X** (*ndarray*) – 2D array containing genotype data with *int8* type
- **np_y** (*ndarray*) – 2D array containing phenotype data with *float* type
- **int_kOfKFold** (*int*) – The k for k-fold cross validation (default: 2)
- **int_nJobs** (*int*) – The number of thread (default: 1)

Returns None

genepi.step5_crossGeneEpistasis_Logistic.**fsigmoid**(*x*, *a*, *b*)

genepi.step5_crossGeneEpistasis_Logistic.**gaussian**(*x*, *mean*, *amplitude*, *standard_deviation*)

2.6.9 step6_ensembleWithCovariates

Created on Feb 2018

@author: Chester (Yu-Chuan Chang)

genepi.step6_ensembleWithCovariates.**ClassifierModelPersistence**(*np_X*, *np_y*, *str_outputFilePath=''*, *int_nJobs=1*)

Dumping ensemble classifier for model persistence

Parameters

- **np_X** (*ndarray*) – 2D array containing genotype data with *int8* type

- **np_y** (*ndarray*) – 2D array containing phenotype data with *float* type
- **str_outputFilePath** (*str*) – File path of output file
- **int_nJobs** (*int*) – The number of thread (default: 1)

Returns None

genepi.step6_ensembleWithCovariates.**EnsembleWithCovariatesClassifier**(*str_inputFileName_feature*, *str_inputFileName_phenotype*, *str_outputFilePath=''*, *int_kOfKFold=2*, *int_nJobs=1*)

A workflow to ensemble genetic features with covariates for L1-regularized Logistic regression.

Parameters

- **str_inputFilePath_feature** (*str*) – File path of input feature files from stage 2 - crossGeneEpistasis
- **str_inputFileName_phenotype** (*str*) – File name of input phenotype data
- **str_outputFilePath** (*str*) – File path of output file
- **int_kOfKFold** (*int*) – The k for k-fold cross validation (default: 2)
- **int_nJobs** (*int*) – The number of thread (default: 1)

Returns

tuple containing:

- float_f1Score_train (float): The F1 score of the model for training set
- float_f1Score_test (float): The F1 score of the model for testing set
- Expected Success Response:

```
"step6: Ensemble with covariates. DONE!"
```

Return type (tuple)

genepi.step6_ensembleWithCovariates.**EnsembleWithCovariatesRegressor**(*str_inputFileName_feature*, *str_inputFileName_phenotype*, *str_outputFilePath=''*, *int_kOfKFold=2*, *int_nJobs=1*)

A workflow to ensemble genetic features with covariates for L1-regularized Lasso regression.

Parameters

- **str_inputFilePath_feature** (*str*) – File path of input feature files from stage 2 - crossGeneEpistasis
- **str_inputFileName_phenotype** (*str*) – File name of input phenotype data
- **str_outputFilePath** (*str*) – File path of output file
- **int_kOfKFold** (*int*) – The k for k-fold cross validation (default: 2)
- **int_nJobs** (*int*) – The number of thread (default: 1)

Returns

tuple containing:

- float_AVG_S_P_train (float): The average of the Peason's and Spearman's correlation of the model for training set
- float_AVG_S_P_test (float): The average of the Peason's and Spearman's correlation of the model for testing set
- Expected Success Response:

```
"step6: Ensemble with covariates. DONE!"
```

Return type (tuple)

genepi.step6_ensembleWithCovariates.**LoadDataForEnsemble**(*str_inputFileName_feature*, *str_inputFileName_phenotype*)

Loading genetic features for ensembling with covariates

Parameters

- **str_inputFilePath_feature** (*str*) – File path of input feature files from stage 2 - crossGeneEpistasis
- **str_inputFileName_phenotype** (*str*) – File name of input phenotype data

Returns

tuple containing:

- np_genotype (ndarray): 2D array containing genotype data with *int8* type
- np_phenotype (ndarray): 2D array containing phenotype data with *float* type

Return type (tuple)

genepi.step6_ensembleWithCovariates.**RegressorModelPersistence**(*np_X*, *np_y*, *str_outputFilePath=''*, *int_nJobs=1*)

Dumping ensemble regressor for model persistence

Parameters

- **np_X** (*ndarray*) – 2D array containing genotype data with *int8* type
- **np_y** (*ndarray*) – 2D array containing phenotype data with *float* type
- **str_outputFilePath** (*str*) – File path of output file
- **int_nJobs** (*int*) – The number of thread (default: 1)

Returns None

2.7 Release History

All notable changes to this project will be documented in this file.

2.7.1 [2.0.10] - 2019-07-29

Added

- Add genome database for intergenic region (hg19, hg38)
- Add functions for modeling intergenic region

- Add warning message if there is no variant past the feature selection
- Add warning message if there is no variant remained in previous step

Fixed

- Update the UCSC mysql host
- Fix error when modeling empty feature set (step5)
- Fix error when modeling empty feature set (step6)
- Refine PRS plot

2.7.2 [2.0.9] - 2019-12-31

Added

- Add standalone AppGenEpi.app

Fixed

- Fix the bug of joblib multithreading

2.7.3 [2.0.8] - 2019-12-26

Fixed

- Fix unsupported image format for ploting PRS

2.7.4 [2.0.7] - 2019-12-26

Fixed

- Fix executable python for AppGenEpi

2.7.5 [2.0.6] - 2019-12-25

Fixed

- Fix file path bug for ploting PRS

Removed

- Remove unused PIL import

2.7.6 [2.0.5] - 2019-12-24

Added

- Add AppGenEpi (GUI)
- Add polygenic risk score calculation for classifier

2.7.7 [2.0.4] - 2019-12-12

Added

- Add sliding windows scanning to deal with mega genes

2.7.8 [2.0.3] - 2019-10-13

Added

- Add GenEpi's documentation to Read the Docs
- Add model persistance for classifier and regressor

2.7.9 [2.0.2] - 2019-09-18

Added

- Add multiprocessing for gene batch running

2.7.10 [2.0.1] - 2019-07-12

Added

- Support self-defiend genome regions (for any species)
- Add argument parser (GenEpi -h)
- Add "GenEpi" to bin when installing
- Add output log
- Add quick test
- Add change log in Changelog.md

Changed

- Change feature encoder for reducing memory usage

Removed

- Remove random forest ensemble

2.7.11 [1.0.3] - 2018-04-22

First preview release

Python Module Index

g

Index

I

L

M

P

R

S