ELPGV Manual

Ensemble learning for integrative prediction of genetic values with genomic variants

Version: 0.1.0

Date: August, 21th 2022

Author: Lin-Lin Gu

Maintainer:

Lin-Lin Gu: linlin-gu@outlook.com

Table of Contents

1 Brief introduction

Whole genome variants offer sufficient information for genetic prediction of human disease, and animal/plant breeding values that has been widely accepted for animal breeding. Many sophisticate statistical algorithms have been developed for enhancing the prediction accuracy; however, each method has its own advantages and disadvantages, so far no method can beat anther. We herein propose a new ensemble learning strategy, called ELPGV (*Ensemble Learning of Prediction for Genetic Value*) for genetic prediction, which assembles predictions from several basic methods, to more accurate predictions.

2 ELPGV function

ELPGV	Ensemble learning for integrative prediction of genetic values

2.1 Description

The ELPGV (*Ensemble Learning of Prediction for Genetic Value*) function is ensemble learning for integrative prediction of genetic values with genomic variants.

2.2 Usage

```
ELPGV(rep_times = 100, interation_times=20, weight_min=0, weight_max=1,
        weight_dimension=ncol(test_PredMat), rate_min=-0.01, rate_max=0.01,
        paticle_number=20, CR = 1.0, train_PredMat=as.matrix(train_PredMat),
        test_PredMat=as.matrix(test_PredMat), R = 0.5, IW = 1, AF1 = 2,
        AF2 = 2, type ="pcc", select="auto", index=NULL)
```

2.3 Arguments

rep_times	Number of repetitions of ELPGV algorithm, default 100.
interation_times	The number of ELPGV iterations within each repetition, default 20.
weight_min	Minimum value of training weight range. The range is from 0 to 1, the default value is 0.
weight_max	Maximum value of training weight range. The range is from 0 to 1, the default value is 1.
weight_dimension	The number of basic learners for ensemble learning.
rate_min	The minimum value of the rate range of particle swarm optimization, the default value is -0.01.

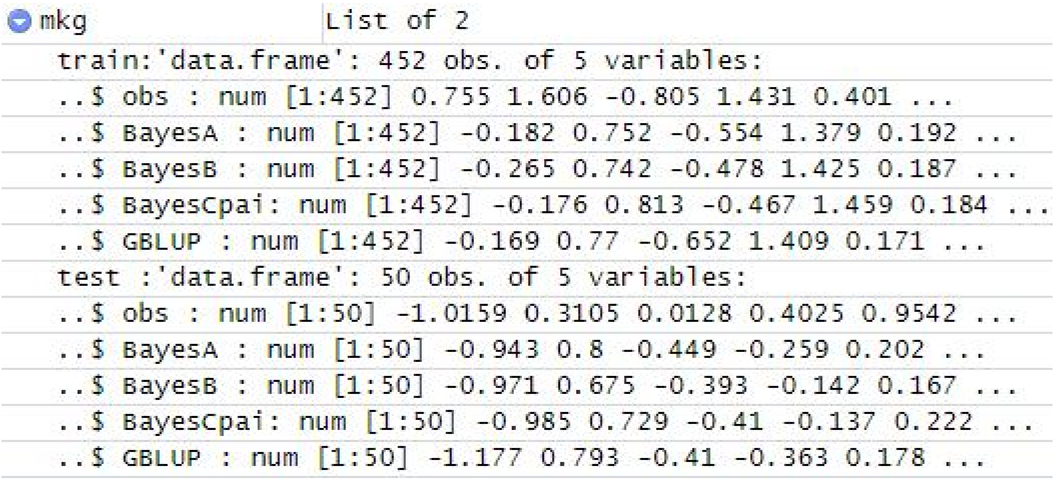
rate_max	The maximum value of the rate range of particle swarm optimization, the default value is 0.01.
paticle_number	Initial population number of particle swarm optimization and differential evolution algorithm. The range is 5N to 10N, where N is the number of basic learners for ensemble learning.
CR	Crossover probability of differential evolution algorithm, the default value is 1.
train_PredMat	Prediction matrix for training group, with Phenotype.
test_PredMat	Prediction matrix for testing group, without Phenotype.
R	Mutation probability of differential evolution algorithm.
IW	Inertia weight of particle swarm optimization, the default value is 0.5.
AF1	The first acceleration factor of particle swarm optimization, the default value is 2.
AF2	The second acceleration factor of particle swarm optimization, the default value is 2.
type	The evaluating indicator, "pcc" is Pearson correlation coefficient.
select	User chooses the way to generate reference phenotype values, setting as either 'auto' or 'manu', if 'auto' is chosen, parameter index (see below) should be chosen accordingly.
index	Indicating which prediction is taken as reference phenotype, f.i., if 3th method (3th columns) is chosen, parameter select='manu' and index=3; otherwise, setting "auto" select: the index is NULL.

2.4 Value

The pred Prediction value of all models.

3 Build in data

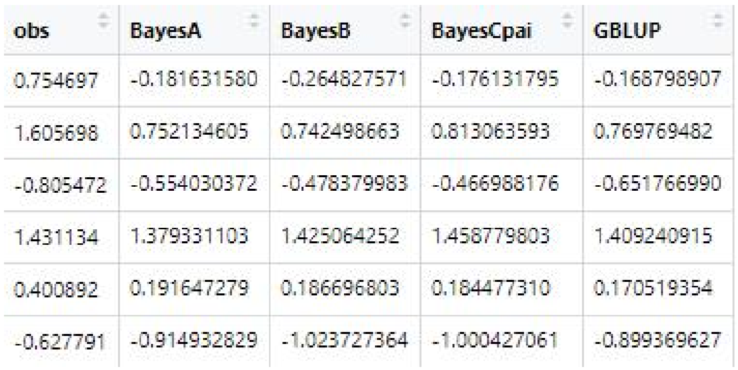
An example datasets 'exdata' is a list that including the milk yield datasets. The mkg list including the prediction matrix of Training set(train) and the prediction matrix of Testing set(test). The train list including the real phenotype value for each individual(obs), the predicted phenotype value of the BayesA model for each individual(BayesA), the predicted phenotype value of the BayesB model for each individual(BayesB), the predicted phenotype value of the BayesCpai model for each individual(BayesCpai) and the predicted phenotype value of the GBLUP model for each individual(GBLUP). The test list structure is the same as the train list, 'exdata' can be loaded with data(exdata).



```
mkg                List of 2
  train:'data.frame': 452 obs. of 5 variables:
  ..$ obs : num [1:452] 0.755 1.606 -0.805 1.431 0.401 ...
  ..$ BayesA : num [1:452] -0.182 0.752 -0.554 1.379 0.192 ...
  ..$ BayesB : num [1:452] -0.265 0.742 -0.478 1.425 0.187 ...
  ..$ BayesCpai: num [1:452] -0.176 0.813 -0.467 1.459 0.184 ...
  ..$ GBLUP : num [1:452] -0.169 0.77 -0.652 1.409 0.171 ...
  test :'data.frame': 50 obs. of 5 variables:
  ..$ obs : num [1:50] -1.0159 0.3105 0.0128 0.4025 0.9542 ...
  ..$ BayesA : num [1:50] -0.943 0.8 -0.449 -0.259 0.202 ...
  ..$ BayesB : num [1:50] -0.971 0.675 -0.393 -0.142 0.167 ...
  ..$ BayesCpai: num [1:50] -0.985 0.729 -0.41 -0.137 0.222 ...
  ..$ GBLUP : num [1:50] -1.177 0.793 -0.41 -0.363 0.178 ...
```

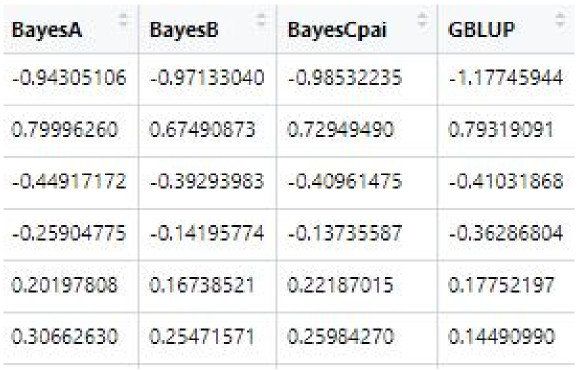
3.1 Running build-in data

```
library("ELPGV")
data(exdata)
train_PredMat = mkg$train
View(train_PredMat)
```



obs	BayesA	BayesB	BayesCpai	GBLUP
0.754697	-0.181631580	-0.264827571	-0.176131795	-0.168798907
1.605698	0.752134605	0.742498663	0.813063593	0.769769482
-0.805472	-0.554030372	-0.478379983	-0.466988176	-0.651766990
1.431134	1.379331103	1.425064252	1.458779803	1.409240915
0.400892	0.191647279	0.186696803	0.184477310	0.170519354
-0.627791	-0.914932829	-1.023727364	-1.000427061	-0.899369627

```
test_PredMat = mkg$test[,-1]
TBV = mkg$test[,1]                 # True breeding value of testing group
View(test_PredMat)
```

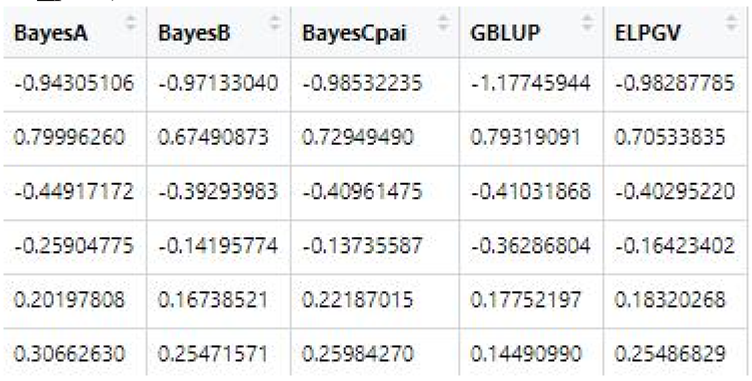


BayesA	BayesB	BayesCpai	GBLUP
-0.94305106	-0.97133040	-0.98532235	-1.17745944
0.79996260	0.67490873	0.72949490	0.79319091
-0.44917172	-0.39293983	-0.40961475	-0.41031868
-0.25904775	-0.14195774	-0.13735587	-0.36286804
0.20197808	0.16738521	0.22187015	0.17752197
0.30662630	0.25471571	0.25984270	0.14490990

```
ELPGV_pred = ELPGV(rep_times = 100, interation_times=20, weight_min=0,
                   weight_max=1, weight_dimension=ncol(test_PredMat),
                   rate_min=-0.01, rate_max=0.01, paticle_number=20,
```

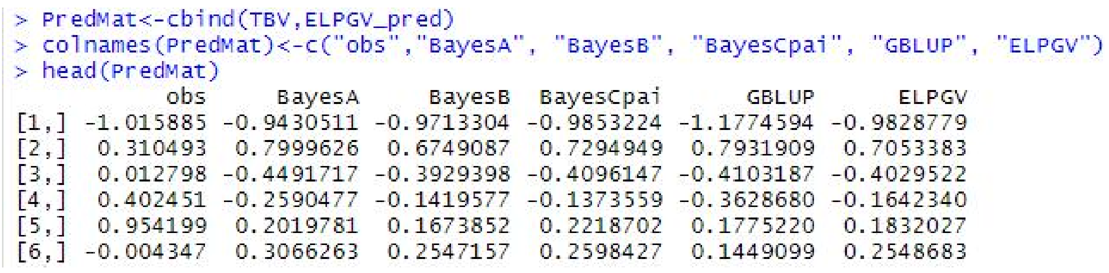
CR = 1.0, train_PredMat=as.matrix(train_PredMat),
test_PredMat=as.matrix(test_PredMat), R = 0.5,
IW = 1, AF1 = 2, AF2 = 2, type ="pcc", select="auto",
index=NULL)

View(ELPGV_pred)



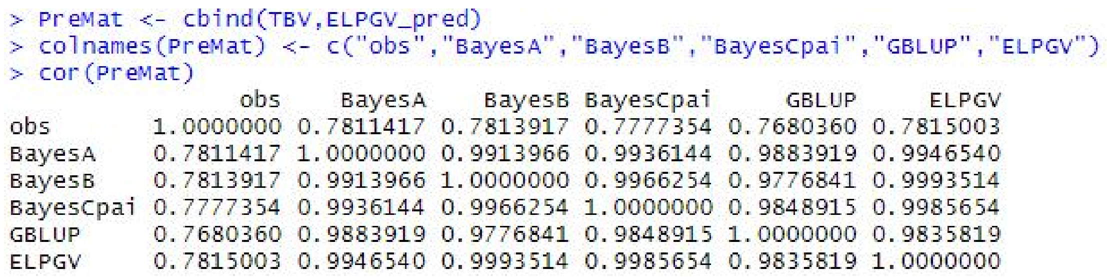
BayesA	BayesB	BayesCpai	GBLUP	ELPGV
-0.94305106	-0.97133040	-0.98532235	-1.17745944	-0.98287785
0.79996260	0.67490873	0.72949490	0.79319091	0.70533835
-0.44917172	-0.39293983	-0.40961475	-0.41031868	-0.40295220
-0.25904775	-0.14195774	-0.13735587	-0.36286804	-0.16423402
0.20197808	0.16738521	0.22187015	0.17752197	0.18320268
0.30662630	0.25471571	0.25984270	0.14490990	0.25486829

PredMat = cbind(TBV,ELPGV_pred)
colnames(PredMat) = c(“obs”, “BayesA”, “BayesB”, “BayesCpai”,
“GBLUP”, “ELPGV”)
head(PredMat)



```
> PredMat<-cbind(TBV,ELPGV_pred)
> colnames(PredMat)<-c("obs","BayesA", "BayesB", "BayesCpai", "GBLUP", "ELPGV")
> head(PredMat)
           obs     BayesA     BayesB  BayesCpai      GBLUP      ELPGV
[1,] -1.015885 -0.9430511 -0.9713304 -0.9853224 -1.1774594 -0.9828779
[2,]  0.310493  0.7999626  0.6749087  0.7294949  0.7931909  0.7053383
[3,]  0.012798 -0.4491717 -0.3929398 -0.4096147 -0.4103187 -0.4029522
[4,]  0.402451 -0.2590477 -0.1419577 -0.1373559 -0.3628680 -0.1642340
[5,]  0.954199  0.2019781  0.1673852  0.2218702  0.1775220  0.1832027
[6,] -0.004347  0.3066263  0.2547157  0.2598427  0.1449099  0.2548683
```

# If the observable value of testing population is given, f.i. in the situation of cross validation, one can evaluates the prediction accuracy by calculating the correlation coefficient between prediction and observable value.



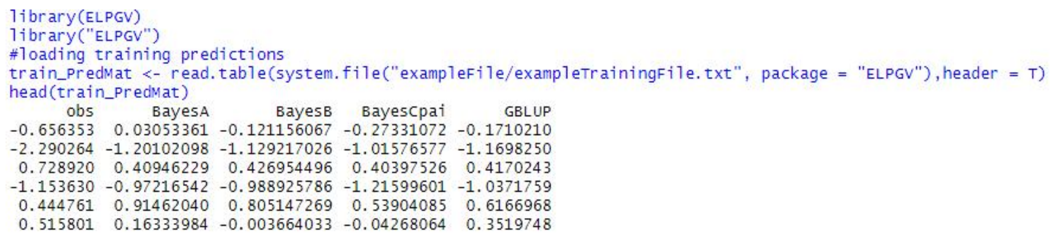
```
> PreMat <- cbind(TBV,ELPGV_pred)
> colnames(PreMat) <- c("obs","BayesA","BayesB","BayesCpai","GBLUP","ELPGV")
> cor(PreMat)
                obs    BayesA    BayesB BayesCpai     GBLUP     ELPGV
obs       1.0000000 0.7811417 0.7813917 0.7777354 0.7680360 0.7815003
BayesA    0.7811417 1.0000000 0.9913966 0.9936144 0.9883919 0.9946540
BayesB    0.7813917 0.9913966 1.0000000 0.9966254 0.9776841 0.9993514
BayesCpai 0.7777354 0.9936144 0.9966254 1.0000000 0.9848915 0.9985654
GBLUP     0.7680360 0.9883919 0.9776841 0.9848915 1.0000000 0.9835819
ELPGV     0.7815003 0.9946540 0.9993514 0.9985654 0.9835819 1.0000000
```

3.2 Quick running your data

We also provide external dataset for testing, which can be accessed from “exa mpleFile” of ELPGV package.

library("ELPGV")
#loading training predictions
train_PredMat = read.table(system.file("exampleFile/*exampleTrainingFile.txt*",
package = "ELPGV"), header = T)

head(train_PredMat)

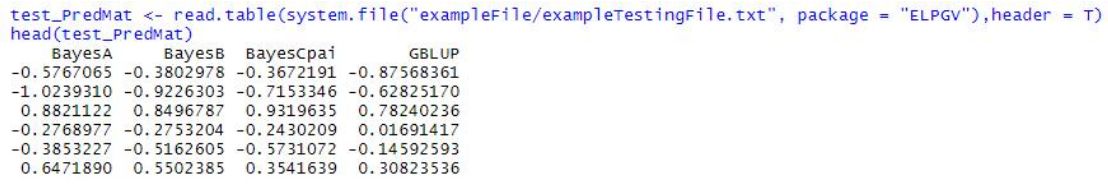


```
library(ELPGV)
library("ELPGV")
#loading training predictions
train_PredMat <- read.table(system.file("exampleFile/exampleTrainingFile.txt", package = "ELPGV"),header = T)
head(train_PredMat)
      obs      BayesA       BayesB   BayesCpai      GBLUP
-0.656353  0.03053361 -0.121156067 -0.27331072 -0.1710210
-2.290264 -1.20102098 -1.129217026 -1.01576577 -1.1698250
 0.728920  0.40946229  0.426954496  0.40397526  0.4170243
-1.153630 -0.97216542 -0.988925786 -1.21599601 -1.0371759
 0.444761  0.91462040  0.805147269  0.53904085  0.6166968
 0.515801  0.16333984 -0.003664033 -0.04268064  0.3519748
```

#loading testing predictions

test_PredMat = read.table(system.file("exampleFile/*exampleTestingFile.txt*",
package = "ELPGV"), header = T)

head(test_PredMat)

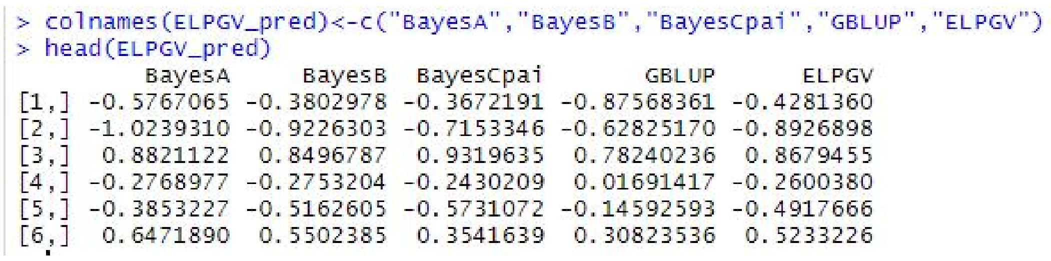


```
test_PredMat <- read.table(system.file("exampleFile/exampleTestingFile.txt", package = "ELPGV"),header = T)
head(test_PredMat)
    BayesA     BayesB  BayesCpai       GBLUP
-0.5767065 -0.3802978 -0.3672191 -0.87568361
-1.0239310 -0.9226303 -0.7153346 -0.62825170
 0.8821122  0.8496787  0.9319635  0.78240236
-0.2768977 -0.2753204 -0.2430209  0.01691417
-0.3853227 -0.5162605 -0.5731072 -0.14592593
 0.6471890  0.5502385  0.3541639  0.30823536
```

ELPGV_pred = ELPGV(rep_times = 100, interation_times=20, weight_min=0,
weight_max=1, weight_dimension=ncol(test_PredMat),
rate_min=-0.01, rate_max=0.01, paticle_number=20,
CR = 1.0, train_PredMat=as.matrix(train_PredMat),
test_PredMat=as.matrix(test_PredMat), R = 0.5,
IW = 1, AF1 = 2, AF2 = 2, type ="pcc", select="auto",
index=NULL)

colnames(ELPGV_pred) = c("BayesA", "BayesB", "BayesCpai",
"GBLUP", "ELPGV")

head(ELPGV_pred)



```
> colnames(ELPGV_pred)<-c("BayesA","BayesB","BayesCpai","GBLUP","ELPGV")
> head(ELPGV_pred)
         BayesA     BayesB  BayesCpai       GBLUP      ELPGV
[1,] -0.5767065 -0.3802978 -0.3672191 -0.87568361 -0.4281360
[2,] -1.0239310 -0.9226303 -0.7153346 -0.62825170 -0.8926898
[3,]  0.8821122  0.8496787  0.9319635  0.78240236  0.8679455
[4,] -0.2768977 -0.2753204 -0.2430209  0.01691417 -0.2600380
[5,] -0.3853227 -0.5162605 -0.5731072 -0.14592593 -0.4917666
[6,]  0.6471890  0.5502385  0.3541639  0.30823536  0.5233226
```

4 Additional explanation of parameter 'select' and 'index'

ELPGV needs to compare the prediction accuracies of training set of different methods first and chooses the best one as reference phenotypic values, by which we are able to evaluate the prediction accuracies for testing set. We herein provide two ways to generate reference phenotype values, (I) auto (automatic) and (II) manu (manual). The "auto" function chooses the best prediction method as reference automatically, whereas the 'manu' function requires users to choose the best one manually, which needs user's experience. 'auto' function is recommended for general users, and "manu" is for advanced users. The example is below, if we need to ensemble four basic models, such as BayesA, BayesB, BayesCπ and GBLUP, we will have the predictions of four methods for training set. Among these predictions, users need to choose the best method with the most accurate predictions as input (reference phenotypic value) for prediction of test set. In this example, the prediction accuracy of BayesCpai is the best and chosen as reference phenotype values. If "auto" function is selected, ELPGV will choose the best method (BayesCpai in this case) automatically (**select="auto"** and **index="NULL**"); but if "manu" function is selected, users can choose any method freely (**select="manu"** and **index=3**, if the third method or the third column is the chosen). It is noted that the performance of ELPGV will rely on the "best method" for training, users should choose the function with caution, if not sure which method is the best, "auto" function is recommended.

I. auto

```
ELPGV(rep_times = 100, interation_times=20, weight_min=0, weight_max=1,
        weight_dimension=ncol(test_PredMat), rate_min=-0.01, rate_max=0.01,
        paticle_number=20, CR = 1.0, train_PredMat=as.matrix(train_PredMat),
        test_PredMat=as.matrix(test_PredMat), R = 0.5, IW = 1, AF1 = 2,
        AF2 = 2, type ="pcc", select="auto", index=NULL)
```

II. manu

```
ELPGV(rep_times = 100, interation_times=20, weight_min=0, weight_max=1,
        weight_dimension=ncol(test_PredMat), rate_min=-0.01, rate_max=0.01,
        paticle_number=20, CR = 1.0, train_PredMat=as.matrix(train_PredMat),
        test_PredMat=as.matrix(test_PredMat), R = 0.5, IW = 1, AF1 = 2,
        AF2 = 2, type ="pcc", select="manu", index=3)
```

5 Code availability

The source code of ELPGV is freely available on request.

6 How to access help

If users have any bugs or issues or any suggestions are available, feel free to contact:
Lin-Lin Gu: linlin-gu@outlook.com