Package 'MAPCUE'

March 23, 2018

Contents

Version 1.0.0

Date 2018-03-22

Title Marker Association with Principle Component Analysis Covaraites and General Linear Model

Depends R(>=3.0)

Imports mgcv

Author Jacob Lamkey, Yuanhong Song

Description MAPCUE is `R` package to perform genome-wide association study (GWAS) by generalized linear model (GLM) method. Besides the core GWAS function, the package includes functions to perform power analysis. The core functions were modified from Zhiwu Zhang, and referenced the ALPACA package by Haixiao Dong and Ryan Oliveira. In MAPCUE packages, we perform each analysis as individual modual to allow maximum flexibility.

Get Started

To use MAPCUE, the user need to install the MAPCUE package first. Upon loading, the package will automatically load the necessary `R` package `mgcv`. The `MAPCUE` package can be loaded to `R` Environment by the following command:

```
source("MAPCUE.R")
library("mgcv")
```

It is recommended to save the MAPCUE `R` file under the same folder as the data, where the folder is set to be the working directory. Upon successful loading of package, five functions will be added into the `R` Environment. If otherwise, please 1) check if the current working directory matches the saving path of the MAPCUE.R file; 2) install the `mgcv` package.

GWASbyGLM

The `GWASbyGLM` is the core function of the MAPCUE package. Based on the genotype and phenotype data provided by the user, the function performs a GWAS analysis by principal analysis and generalized linear regression model method:

$$Y \sim PC_{1:k} + SNP_{1:p} + \epsilon$$

where $PC_{1:k}$ is the first k principal components, and SNP_{1:p} is the p SNP's provided by user. Given a set of genotype and phenotype data, the GWASbyGLM function first perform a PCA on the genotype. Based on the choice of k, the first k principal component will be used to form an predictor matrix with SNP's. Then a linear model is fitted by least square method. For each SNP, we test for $_H_0$: $\beta_i = 0$ and collect the p value. The function returns the p values of all SNP's.

Inputs

geno: the genotype data (SNP). An $n \times p$ matrix or dataframe, where n is the number of observation and p is the number of SNP's. The geno data should be numerical without missing values. If missing values appears in the data, please consider imputate the data before performing GWASbyGLM.

pheno: the phenotype data. An $n \times 1$ vector or data array, where n is the number of observation. The pheno data should be numerical without missing values. If missing values appears in the data, please consider imputate the data before performing GWASbyGLM. `pheno` can be a simulated data.

CV: the covariates provided by the user with the first columnes indicates the individual's lable. The number of observation needs to match the number of individuals in the geno dataset, and the number of columnes can vary. The data should be numeric.

PCA.M: a single value integer indicates the number of principal components to be included in the GLM model.

Outputs

P: the p value for the test $_H_0$: $\beta_i = 0$ for each SNP. P is a $1 \times p$ number array.

Example:

Example of the input data:

```
# An example of the geno data - a 281 by 3093 matrix
# only a portion of the data is shown
geno[1:5, 1:6]
```

```
##    taxa PZB00859.1 PZA01271.1 PZA03613.2 PZA03613.1 PZA03614.2
## 1 33-16          2          0          0          2          2
## 2 38-11          2          2          0          2          2
## 3  4226          2          0          0          2          2
## 4  4722          2          2          0          2          2
## 5  A188          0          0          0          2          2
```

```
# An example of the pheno data - a 281 by 1 vector
pheno[1:5]
```

```
## [1] 5.008458 5.959332 9.858454 4.590149 7.879969
```

```r
# An example of the CV data - a matrix with 281 rows
CV[1:5,]
```

```
##    Taxa  FactorA  FactorB
## 1 33-16 2.531331 5.501464
## 2 38-11 2.633860 4.655691
## 3  4226 1.890695 6.136883
## 4  4722 1.856035 7.841858
## 5  A188 2.552629 5.409450
```

```r
# An example of the PCA.M value
PCA.M
```

```
## [1] 3
```

To process the command:

```r
# Use a object P to collect the result of the GWAS by GLM function:
P=GWASbyGLM(geno=geno, pheno=pheno,CV=CV,PCA.M=3)
P[1:5]
```

```
## [1] 0.1284659 0.5734759 0.6605255 0.2658064 0.5336817
```

---

manh.plot

The `manh.plot` creates a Manhattan plot which plots the negative log p value against the SNP position. The P values can be the output of the GWASbyGLM function or other result of a GWAS analysis.

Inputs

P an array of p values. The data should be numeric which ranges from 0 to 1.

GM the genometype head information provided by the user. The data set should contains a columne named "Chromosome", which provides the chromosome information for the SNP's.

cutoff a single valued overall error rate choosed by the user, ranges from 0 to 1.

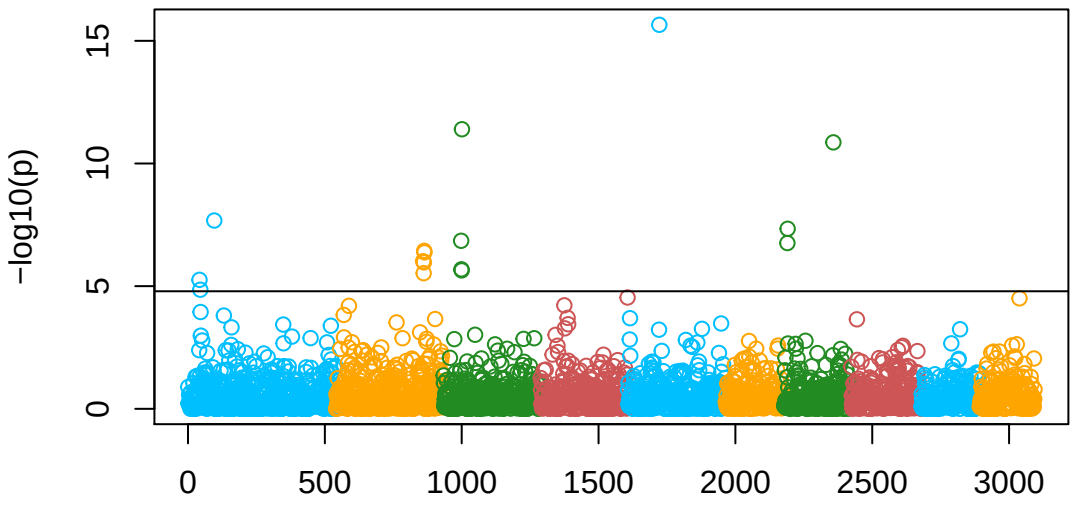
QTN.position (optional) the QTN position.

Outputs

The function returns a Manhattan plot. If the QTN position is provided, the true QTN is labeled by a circle.

Example

```r
manh.plot(P=P,GM=GM, cutoff=.05)
```



---

qqGLM

This function assumes the null p values (the GWAS p value for the non-QTN SNP's) are distributed uniformally, and creates a negative log P quantile-quantile plot for the GWAS p values against the null p value.

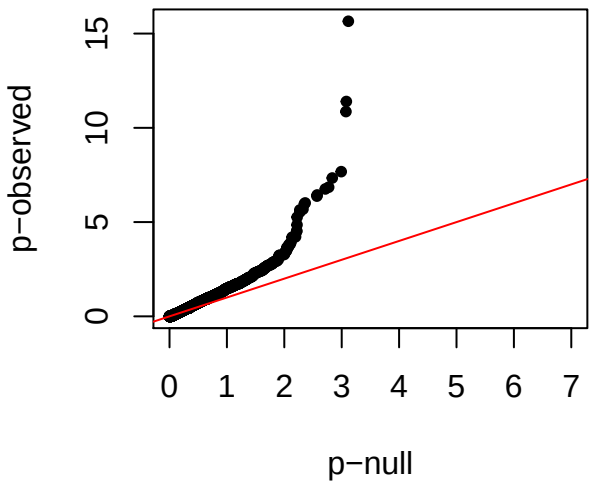
Inputs

P an array of p values. The data should be numeric which ranges from 0 to 1.

Outputs

This function creates a QQ plot.

```
qqGLM(P)
```



---

fal.pos

With the cutoff significance at the user's choice, the `fal.pos` function identify the number of false positive. To perform this analysis, user needs to provide the true QTN position.

Inputs

P an array of p values. The data should be numeric which ranges from 0 to 1.

QTN.position the QTN position.

cutoff a single valued overall error rate choosed by the user, ranges from 0 to 1.

Outputs

This function returns the number of false positive.

Exampel

```
fal.pos(P, QTN.position, 0.05)
```

```
## [1] 15
```

---

fal.neg

Inputs

P an array of p values. The data should be numeric which ranges from 0 to 1.

QTN.position the QTN position.

cutoff a single valued overall error rate choosed by the user, ranges from 0 to 1.

Outputs

This function returns the number of false negative.

Exampel

```
fal.neg(P, QTN.position, 0.05)
```

```
## [1] 9
```