Package: CIVMR (via r-universe)

October 4, 2024

Title Constrained Instrumental Variables in Mendelian Randomization with Pleiotropy

Version 0.0.1

Description In Mendelian randomization (MR), genetic variants are used to construct instrumental variables that then enable inference about the causal relationship between a phenotype of interest and a response or disease outcome. However, valid MR inference requires several assumptions, including the assumption that the genetic variants only influence the response through the phenotype of interest.Pleiotropy occurs when a genetic variant has an effect on more than one different phenotypes, and therefore a pleiotropic genetic variant may be an invalid instrumental variable.Hence, a naive method for constructing an instrumental variables may lead to biased estimation of the association between the phenotype and the response. Here, we encode a new and intuitive method (Constrained Instrumental Variable method [CIV]) to construct valid instrumental variables and perform adjusted causal effect estimation when pleiotropic exists, focusing particularly on the situation where pleiotropic phenotypes have been measured. Our approach is theoretically guaranteed to perform an automatic and valid selection of genetic variants when building the instrumental variable. We also provide details of the features of many existing methods, together with a comparison of their performance in a large series of simulations. CIV performs robustly across many different pleiotropic violations of the MR assumptions.

Depends R (>= 3.0.0)

Suggests PMA, plyr, glmnet, Matrix, foreach, caret, ggplot2, sem, MASS, knitr, rmarkdown

License What license is it under?

Encoding UTF-8

LazyData true

RoxygenNote 5.0.1

VignetteBuilder knitr

Repository https://mrcieu.r-universe.dev

RemoteUrl https://github.com/LaiJiang/CIVMR

RemoteRef HEAD

RemoteSha 972613ed33bc61560a0972ef995c405e7f9c3118

Contents

ADNI *ADNI Data.*

Description

A dataset collected from ADNI project for MR analysis in the CIV paper. It contains 491 subjects, whose phenotypes ($A\beta$, Ptau, Ttau, Glucose levels) and Alzheimer's status were collected. 20 SNPs associated with $A\beta$ were selected and their dosage of the 491 subjects were recorded. This data file is extracted to serve as an example to estimate the causal effect of $A\beta$ on progression of Alzheimer's disease while accounting for potential pleiotropic effect from other phenotypes.

Usage

```
data(ADNI)
```

Arguments

ADNI$Y: the Alzheimer's disease status. Continuous variable. The raw status is binary variable, and we adjusted it for confounding factors such as sex, age, education ... etc.

ADNI$X: The phenotype of interest $A\beta$ Continuous variable.

ADNI$Z: The potential pleiotropic phenotypes (Ptau, Ttau, Glucose levels). Continuous variables.

ADNI$G: genotypes. The adjusted dosage of 20 SNPs.

Format

An object of class "data.frame".

Examples

```
data(ADNI)
X <- ADNI$X
Z <- ADNI$Z
G <- ADNI$G
Y <- ADNI$Y
```

allele *cross-validated Allele score method.*

Description

This function implement Allele score methods with cross-validation in the way Stephen Burgess suggested in the Allele score methods paper.

Usage

```
allele(MR.data, n_folds = 10)
```

Arguments

MR.data: data frame containing G,X,Z,Y.

n_folds: the number of folds for cross-validation.

Value

weights: the weights for allele score across folds. Each row is a weight vector corresponding to a specific fold.

allele_score: The cross-validated Allele score, which would be used as the new instruments in MR analysis.

beta_est: the causal effect estimation of β.

Examples

```
data(simulation)
allele.score <- allele(simulation,n_folds=10)
```

`boot_CIV` *bootstrapped CIV (recommended).*

Description

This function generate a bootstrapped CIV w/wo correction. Specifically, for a bootstrap sample we can generate civ solution u. The boostrap corrected solution u is obtained as the global solution u - (bootrapped average u - global u).

Usage

```
boot_CIV(MR.data, n_boots = 10)
```

Arguments

`MR.data:`	a data frame containing G,X,Z,Y.
`n_boots:`	number of bootstrap samples.

Value

boots.u: bootstrapped CIV solution u (without correction).

boots.cor.u: bootstrap corrected solution of u. (suggested)

Examples

```
data(simulation)
boot.civ <- boot_CIV(simulation)
#plot the bootstrap corrected solution u.
plot(boot.civ$boots.cor.u)
```

`CIV` *Find a unique solution of CIV.*

Description

This function find the unique CIV solution.

Usage

```
CIV(MR.data)
```

Arguments

`MR.data:`	A data.frame() object containg G,X,Z,Y.

Value

c: solution vector to the constrained maximization problem.

max_value: the maximized correlation value.

CIV: the new instrumentable variable for MR analysis: constrained instrumental variable.

Examples

```
data(simulation)
civ.fit <- CIV(simulation)
#plot the weight c
plot(civ.fit$c)
```

`cv_CIV` *cross-validated CIV.*

Description

This function produce a Constrained Instrumental Variable with cross-validation. Specifically, for a predefined fold the CIV is calculated using all samples except this fold, then the CIV solution is applied to the samples in this fold to obtain corresponding CIV. In this way the correlation between samples are expected to be reduced.

Usage

```
cv_CIV(MR.data, n_folds = 10)
```

Arguments

`MR.data:`	a data frame containing G,X,Z,Y.
`n_folds:`	number of folds for cross-validation.

Value

weights: A matrix with dimension $n_folds * p$. Each row is a CIV solution c from a specific fold.

civ.IV: cross-validated CIV instrument $G^* = Gc$.

beta_est: causal effect estimation of X on Y using CIV instrument civ.IV

Examples

```
data(simulation)
cv.civ <- cv_CIV(simulation)
#strong correlation between CIV solutions from different folds.
cor(t(cv.civ$weights))
```

`IV_reduction` *Instrumental variable reduction.*

Description

This function remove highly correlated IVs. An upgraded function SNP_reduction() is suggested.

Usage

```
IV_reduction(snp_matrix, crit_high_cor = 0.8)
```

Arguments

`snp_matrix:`	IV matrix with dimension n * p.
`crit_high_cor:`	criteria to choose highly correlated SNPs. default is 0.8 correlation.

Value

sel_snp: the selected IVs.

id_snp: the ids (columns) of selected IVs in the original IV matrix.

Examples

```
data(simulation)
snp.rdc <- IV_reduction(simulation$G)
```

`LA_decomposition` *linear algebra decompositions for CIV. (internal function.)*

Description

This function implements linear algebra steps to acquire necessary matrices for CIV construction.

Usage

```
LA_decomposition(G, X, Z)
```

Arguments

`G:`	original instruments with dimension nXp.
`X:`	phenotype of interest. dimension nXk.
`Z:`	possible pleiotropic phenotypes which have been measured. dimension nXr.

Value

A list of matrices which will be called by solv_pcc() and pcc_IV().

Examples

```
data(simulation)
LA_decomposition(simulation$G,simulation$X,simulation$Z)
```

`lmp` *simple linear regression pvalues (internal function.)*

Description

univaraite t-test pvalues for a regression.

Usage

```
lmp(modelobject)
```

Arguments

`modelobject:`	a regression object.

Value

p: pvalue.

`lmPvalue` *univariate T-test p-values.*

Description

Given response Y and a set of features X, this function obtains univariate T-test p-values for each of the feature for selection purpose.

Usage

```
lmPvalue(Y, X)
```

Arguments

`Y:`	response variable. $n \times 1$.
`X:`	the independent features.$n \times p$.

Value

pvalue_list: the list of Pvalues for feature selection based on univariate T-test.

Examples

```
data(simulation)
p.values <- lmPvalue(simulation$X, simulation$G)
```

pcc_IV *multiple orthogonal CIV solutions. (internal function)*

Description

This function find multiple CIV solutions that are orthogonal to each other. Only the first one achive the global maximum correlation.

Usage

```
pcc_IV(A, B, G, inv_GG_square, no_IV = ncol(G) - ncol(B))
```

Arguments

`A:`	matrix given by LA_decomposition().
`B:`	matrix given by LA_decomposition().
`G:`	original instruments.

Value

u_max: the solution of u that would maximize the constrained correlation problem.

Examples

```
data(simulation)
#CIV linear algebra decomposition components
civ.deco <- LA_decomposition(simulation$G,simulation$X,simulation$Z)
#solve the CIV solution for c
civ.mult <- pcc_IV(civ.deco$A, civ.deco$B, simulation$G, civ.deco$inv_GG_square)
```

rm_outlier_IV *select IVs from a smooth_IV object (experimental function).*

Description

this function removes IVs with extreme low correlation and extreme high prediction error. This is an experimental function to check how many redundant solutions are found in smooth.opt object.

Usage

```
rm_outlier_IV(smooth_IV, MR.data, crit = 0.9, sigma_min = 0.01)
```

Arguments

`smooth_IV:`	an object from smooth_CIV() function.
`MR.data:`	data frame containing G,X,Z,Y.
`......:`	default values for other tuning parameters.

Value

IV_mat: the final matrix of CIV instruments.

u_mat: the final CIV solutions of u. Each column is a distinct solution.

Examples

```
data(simulation)
G <- simulation$G
X <- simulation$X
Z <- simulation$Z
Y <- simulation$Y
smooth.opt <- smooth_CIV( G,X,Z,Y, k_folds = 10)
smooth.clean <- rm_outlier_IV(smooth.opt, simulation)
dim(smooth.clean$u_mat) #check how many solutions are different. It is probability much less than 100.
```

`simulation` *simulation Data.*

Description

A simulated data using the same framework from simulation series I in the CIV paper. It contains 500 subjects with one phenotype (X) of interest, one pleiotropic phenotype (Z) and one outcome (Y) simulated for each subject. 9 SNPs were generated and they were associated with both X and Z. The true causal effect of X on Y is 1. This data file is simulated to serve as an example to estimate the causal effect of X on Y while accounting for potential pleiotropic effect from Z. Users can compare the performance of different MR methods on this simulation dataset since the true causal effect is known.

Usage

```
data(simulation)
```

Arguments

`simulation$Y:` the simulated outcome Y. Continuous variable.

`simulation$X:` The simulated phenotype of interest X. Continuous variable.

`simulation$Z:` The potential pleiotropic phenotype Z. Continuous variable.

`simulation$G:` The simulated genotypes. The dosage of 9 independent SNP variants were simulated with a minor allele frequency of 0.3 for all 500 subjects.

Format

An object of class `"data.frame"`.

Examples

```
data(simulation)
X <- simulation$X
Z <- simulation$Z
geno <- simulation$G
outcome <- simulation$Y
```

`smooth_CIV` *CIV_smooth solution with cross-validation.(recommended)*

Description

This function first find the optimal value of λ according to projected prediction error with cross-validation. Then for a given λ value multiple intial points are used to explore potentially multiple modes.

Usage

```
smooth_CIV(G, X, Z, Y, lambda_list = NULL, k_folds = 10, sigma_min = 0.01,
  sigma_up = 0.5, stepsize = 0.1, conv_iters = 5, stepsize_last = 1e-04,
  last_conv_iters = 2000, method_lambda = "er", n_IV = 100)
```

Arguments

`initial:` the initial value for updating u.

`G:` SNP matrix with dimension $n \times p$.

`X:` phenotype of interest.

`Z:` pleiotropic phenotype Z.

`Y:` the disease outcome Y.

`lambda_list:` a list of values for regularization parameter lambda. A default list will be chosen if not provided.

`k_folds:` number of folds for cross-validation (to find optimum λ). default = 10.

`n_IV:` the number of initial points chosen to explore potential multiple modes. The converged solutions will be screened to delete redundant solutions. So the final solutions will be less or equal to n_IV. default = 100.

`sigma_min:` the minimum value of σ (corresponding to the closeast approximation of L_0 penalty). default = 0.01.

`sigma_up:` the moving down multiplier. $\sigma_{j+1} = sigma_u p \times \sigma_j$. default = 0.5.

`stepsize:` the stepsize to move solution u. default = 0.1.

`conv_iters:` the maximum steps to allow updating when a converged solution is found. default =5.

`stepsize_last:` When a converged solution is found with stepsize, we update this solution with a smaller stepsize to achive a more precise local maximum solution. default = 0.0001.

`last_conv_iters`: the maximum iterations to run in the stage of “refining" optimum solution. default = 2000.

`......`: default values for other tuning parameters.

Value

opt_lambda: the chosen optimum value of λ corresponding to the minimum projected prediction error (see paper).

IV_mat: the final matrix of CIV instruments with respect to the opt_lambda. Each column is a new instrument.

u_mat: the final CIV solutions of u with respect to the opt_lambda. Each column is a converged solution.

G_pred_error_list: the projected prediction error according to the list values of λ.

Pred_error_list: the prediction error according to the list values of λ.

Examples

```
data(simulation)
G <- simulation$G
X <- simulation$X
Z <- simulation$Z
Y <- simulation$Y
smooth.opt <- smooth_CIV( G,X,Z,Y, k_folds = 10)
plot(smooth.opt$u_mat[,1])  #plot a solution u.
```

`smooth_L0_lambda` *CIV_smooth solution given λ. (Internal function)*

Description

This function finds a CIV_smooth solution of u given a value of λ. This function is mostly for internal use. smooth_CIV() is suggested for users to obtain optimal solutions of CIV_smooth.

Usage

```
smooth_L0_lambda(initial = NULL, null_space, G, X, GTG, lambda,
  sigma_min = 0.01, sigma_up = 0.5, stepsize = 0.1, conv_iters = 5,
  stepsize_last = 1e-04, last_conv_iters = 2000, GTMG, ZTG, GTZ, ZTG_ginv,
  accuracy_par = 1e-10)
```

Arguments

`initial:` the initial point of u for updating. The CIV solution will be used as the initial point if no choice is made.

`G:` SNP matrix with dimension $n \times p$.

`X:` phenotype of interest.

`Z:` pleiotropic phenotype Z.

`GTG:` $G‘G$

`GTMG:` $G‘X(X‘X)^{-1}X‘G$.

`ZTG:` $Z‘G$

`GTZ:` $G‘Z$

`ZTG_ginv:` general inverse of $Z‘G$ (ginv($Z‘G$)).

`null_space:` null space of matrices G‘Z (null(G‘Z)).

`lambda:` a given value (must be specified) for regularization parameter λ.

`accuracy_par:` the accuracy threshold parameter to determine if the algorithm converged to a local maximum. Default is 1e-10.

`last_conv_iters:`
the maximum iterations to run. Default is 2000.

`......:` default values for other tuning parameters.

Value

mat_u: the trace of all updated iterations of u.

opt_solution: the final solution of u.

value_list: the iteration values of target function (penalized correlation).

unstrained_val_list: the iteration values of correlation between X and Gu.

dev_list: the iteration values of deviance between updated vector of u.

n_iters_stage: the number of iterations before finishing updating. If this value < last_conv_iters, then the algorithm stopped at a solution of u without using up its updating quota.

sigma_stage: the updating values of σ that are used in each iteration.

stepsize_list: the updating values of stepsize that are used in each iteration.

Examples

```
data(simulation)
G <- simulation$G
X <- simulation$X
Z <- simulation$Z
GTG <- crossprod(G,G)
M <- tcrossprod ( tcrossprod ( X , solve(crossprod(X,X) ) ), X )
GTMG <- crossprod(G, crossprod(M,G))
ZTG <- crossprod(Z,G)
GTZ <- crossprod(G,Z)
null_space <- Null( GTZ)
```

```
ZTG_ginv <- ginv(ZTG)
lambda <- 1
smooth.lambda1 <- smooth_L0_lambda(null_space = null_space, G = G, X = X, GTG = GTG, lambda = lambda,
GTMG = GTMG, ZTG = ZTG, GTZ = GTZ, ZTG_ginv = ZTG_ginv )
plot(smooth.lambda1$opt_solution)  #plot the final solution u
```

SNP_reduction *SNP pre-processing.*

Description

This function remove highly correlated SNPs. It also calculate MAF for each snp and delete rare snps with low MAF (e.g. 0.01).

Usage

```
SNP_reduction(snp_matrix, crit_high_cor = 0.8, maf_crit = 0.01)
```

Arguments

`snp_matrix:`	SNP matrix with dimension n * p.
`crit_high_cor:`	criteria to choose highly correlated SNPs.
`maf_crit:`	criteria to choose rare SNPs.

Value

sel_snp: the new dosage matrix of selected SNPs.

id_snp: the ids (columns) of selected SNPs in the original SNP matrix.

Examples

```
data(simulation)
snp.rdc <- SNP_reduction(simulation$G)
```

solve_pcc *Find a unique solution of CIV (internal use).*

Description

This function find a unique solutin to the constrained instrument problem given matrix A and B.

Usage

```
solve_pcc(A, B)
```

Arguments

`A:` matrix given by LA_decomposition().

`B:` matrix given by LA_decomposition().

Value

c: solution to the constrained maximization problem.

max_value: the maximized correlation value.

Examples

```
data(simulation)
#CIV linear algebra decomposition components
civ.deco <- LA_decomposition(simulation$G,simulation$X,simulation$Z)
#solve the CIV solution \eqn{u}
civ.c <- solve_pcc(civ.deco$A, civ.deco$B)
#plot the weight c
plot(civ.c$c)
```

TSLS_IV *Two stage least square method.*

Description

This function implement ordinary two stage least square regression and provide variance estimation (if requested).

Usage

```
TSLS_IV(MR.data, Fstats = FALSE, var_cal = FALSE)
```

Arguments

`MR.data:` data frame containing G,X,Z,Y.

`Fstats:` return F-statistics or not. If multiple phenotypes (X) are used, Pillai statistics will be used instead.

`var_cal:` return variance estimation or not.

Value

coef: the causal effect estimation β.

var: the variance estimation of β. if var_cal=TRUE.

stats: F-statistics (or Pillai statistics). if Fstats=TRUE.

pvalue: the pvalue of F-statistics. if Fstats=TRUE.

Examples

```
data(simulation)
TSLS_IV(simulation,Fstats=TRUE,var_cal=TRUE)
```

Index