Package 'SFSI'

March 10, 2022

Title Sparse Family and Selection Index

Version 1.1.0

Date 2022-03-10

Description Here we provide tools for the estimation of coefficients in penalized regressions when the (co)variance matrix of predictors and the covariance vector between predictors and response, are provided. These methods are extended to the context of a Selection Index (commonly used for breeding value prediction). The approaches offer opportunities such as the integration of high-throughput traits in genetic evaluations ('Lopez-Cruz et al., 2020') <doi:10.1038/s41598-020-65011-2> and solutions for training set optimization in Genomic Prediction ('Lopez-Cruz & de los Campos, 2021') <doi:10.1093/genetics/iyab030>.

LazyLoad true

Depends R (>= 3.5)

Imports stats

Suggests BGLR, float, knitr, rmarkdown, ggplot2, parallel, reshape2, viridis, igraph

LinkingTo float

VignetteBuilder knitr

Encoding UTF-8

License GPL-3

NeedsCompilation yes

Author Marco Lopez-Cruz [aut, cre],
Gustavo de los Campos [aut],
Paulino Perez-Rodriguez [ctb]

Maintainer Marco Lopez-Cruz <maraloc@gmail.com>

Repository CRAN

Date/Publication 2022-03-10 21:30:02 UTC

R topics documented:

`BinaryFiles` *Binary files*

Description

Save/read a numeric data as a fortran-formatted binary file at a defined precision (single or double).

Usage

```
saveBinary(X, file = paste0(tempdir(), "/file.bin"),
 type = c("float","double"), verbose = TRUE)

readBinary(file = paste0(tempdir(), "/file.bin"),
  index.row = NULL, index.col = NULL, verbose = TRUE)
```

Arguments

`X` (numeric matrix) Data to save

`file` (character) Name of the binary file to save/read

`type` (character) Either 'float' or 'double' for single (4 bytes) or double precision (8 bytes), respectively, that matrix to save will occupy

`index.row` (integer vector) Which rows are to be read from the file. Default `index.row=NULL` will read all the rows

`index.col` (integer vector) Which columns are to be read from the file. Default `index.col=NULL` will read all the columns

`verbose` `TRUE` or `FALSE` to whether printing file information

Value

Function 'saveBinary' does not return any value but print a description of the file saved.

Function 'readBinary' returns the data that was read.

Author(s)

Marco Lopez-Cruz (<maraloc@gmail.com>) and Gustavo de los Campos

Examples

```
require(SFSI)

# Simulate matrix
X = matrix(rnorm(5000),ncol=5)
head(X)

# Save matrix
saveBinary(X,paste0(tempdir(),"/Matrix1.bin"),type="double") # as double-precision
saveBinary(X,paste0(tempdir(),"/Matrix2.bin"),type="float")  # as single-precision

# Read the double-precision matrix
X2 = readBinary(paste0(tempdir(),"/Matrix1.bin"))
head(X2)
max(abs(X-X2))     # No loss of precision
object.size(X2)    # Size of the object

# Read the single-precision matrix
X3 = readBinary(paste0(tempdir(),"/Matrix2.bin"))
head(X3)
max(abs(X-X3))     # Loss of precision
object.size(X3)    # But smaller-size object

# Read specific rows and columns
index.row = c(2,4,5,8,10)
index.col = c(1,2,5)
X2 = readBinary(paste0(tempdir(),"/Matrix1.bin"),index.row=index.row,index.col=index.col)
X2
# Equal to:
X[index.row,index.col]
```

`collect` *collect function*

Description

Collects all outputs saved at the provided `saveAt` parameter from the SSI analysis when testing data was splited according to argument `subset`.

Usage

```
collect(prefix = "")
```

Arguments

`prefix`	(character) Prefix that was added to the output files name, this may include a path

Value

An object of the class 'SSI' for which methods `fitted`, `plot` and `summary` exist

Author(s)

Marco Lopez-Cruz (<maraloc@gmail.com>) and Gustavo de los Campos

Examples

```
require(SFSI)
data(wheatHTP)

index = which(Y$CV == 1)             # Use only a subset of data
M = scale(M[index,])/sqrt(ncol(M))   # Subset and scale markers
G = tcrossprod(M)                    # Genomic relationship matrix
y = as.vector(scale(Y[index,"E1"]))  # Subset response variable

prefix <- paste0(tempdir(),"/testSSI")

# Run the analysis into 4 subsets and save them at a given prefix
fm <- SSI(y,K=G,tst=1:80,trn=81:length(y),subset=c(1,4),save.at=prefix)
fm <- SSI(y,K=G,tst=1:80,trn=81:length(y),subset=c(2,4),save.at=prefix)
fm <- SSI(y,K=G,tst=1:80,trn=81:length(y),subset=c(3,4),save.at=prefix)
fm <- SSI(y,K=G,tst=1:80,trn=81:length(y),subset=c(4,4),save.at=prefix)

# Collect all results after completion
fm <- collect(prefix)
```

covariance_matrix *Conversion of a covariance matrix to a distance or correlation matrix*

Description

Computes a correlation matrix or a Euclidean distance matrix from a covariance matrix

Usage

```
cov2dist(V, void = FALSE)

cov2cor2(V, a = 1, void = FALSE)
```

Arguments

`V`	(numeric matrix) Symmetric variance-covariance matrix among p variables. It can be of the "float32" type as per the 'float' R-package
`void`	`TRUE` or `FALSE` to whether return or not return the output. When `FALSE` no result is displayed but the input `V` is modified. Default `void=FALSE`
`a`	(numeric) A number to multiply the whole resulting matrix by. Default `a=1`

Details

For any variables X_i and X_j with mean zero and with sample vectors $\mathbf{x}_i = (x_{i1},...,x_{in})'$ and $\mathbf{x}_j = (x_{j1},...,x_{jn})'$, their (sample) variances are equal (up-to a constant) to their cross-products, this is, $var(X_i) = \mathbf{x}_i'\mathbf{x}_i$ and $var(X_j) = \mathbf{x}_j'\mathbf{x}_j$. Likewise, the covariance is $cov(X_i,X_j) = \mathbf{x}_i'\mathbf{x}_j$.

Distance. The square of the distance $d(X_i,X_j)$ between the variables expressed in terms of cross-products is

$$d^2(X_i,X_j) = \mathbf{x}_i'\mathbf{x}_i + \mathbf{x}_j'\mathbf{x}_j - 2\mathbf{x}_i'\mathbf{x}_j$$

Therefore, the output (square) distance matrix will contain as off-diagonal entries

$$d^2(X_i,X_j) = var(X_i) + var(X_j) - 2cov(X_i,X_j)$$

while in the diagonal, the distance between one variable with itself is $d^2(X_i,X_i) = 0$

Correlation. The correlation between the variables is obtained from variances and covariances as

$$cor(X_i,X_j) = cov(X_i,X_j)/(sd(X_i)sd(X_j))$$

where $sd(X_i) = \sqrt{var(X_i)}$; while in the diagonal, the correlation between one variable with itself is $cor(X_i,X_i) = 1$

Variances are obtained from the diagonal values while covariances are obtained from the out-diagonal.

Value

Function 'cov2dist' returns a matrix containing the (square) Euclidean distances. Function 'cov2cor2' returns a correlation matrix

Author(s)

Marco Lopez-Cruz (<`maraloc@gmail.com`>) and Gustavo de los Campos

Examples

```
require(SFSI)
data(wheatHTP)

X = scale(Y[,4:7])
(V = crossprod(X))             # Covariance matrix

# Covariance matrix to distance matrix
(D1 = cov2dist(V))
# it must equal (but faster) to:
D0 = as.matrix(dist(t(X)))^2
max(abs(D0-D1))

# Covariance to a correlation matrix
(R1 = cov2cor2(V))
# it must equal (but faster) to:
R0 = cov2cor(V)
max(abs(R0-R1))

if(requireNamespace("float")){
 # Using a 'float' type variable
 V2 = float::fl(V)
 D2 = cov2dist(V2)
 max(abs(D1-D2))   # discrepancy with previous matrix
 R2 = cov2cor2(V2)
 max(abs(R1-R2))   # discrepancy with previous matrix
}

# Using void=TRUE
cov2dist(V,void=TRUE)
V       # notice that V was modified
cov2dist(V2,void=TRUE)
V2       # notice that V2 was modified
```

fitBLUP *Function fitBLUP*

Description

Solves the Linear Mixed Model and calculates the Best Linear Unbiased Predictor (BLUP)

Usage

```
fitBLUP(y, X = NULL, Z = NULL, K = NULL, U = NULL,
        d = NULL, theta = NULL, BLUP = TRUE,
        method = c("REML","ML"), return.Hinv = FALSE,
        tol = 1E-5, maxiter = 1000, interval = c(1E-9,1E9),
        warn = TRUE)
```

Arguments

`y` (numeric vector) Response variable

`X` (numeric matrix) Design matrix for fixed effects. When `X=NULL` a vector of ones is constructed only for the intercept (default)

`Z` (numeric matrix) Design matrix for random effects. When `Z=NULL` an identity matrix is considered (default) thus **G = K**; otherwise $\mathbf{G} = \mathbf{Z}\ \mathbf{K}\ \mathbf{Z'}$ is used

`K` (numeric matrix) Kinship relationships. This can be of the "float32" type as per the 'float' R-package, or a (character) name of a binary file where the matrix is stored

`U` (numeric matrix) Eigenvectors from spectral value decomposition of $\mathbf{G} = \mathbf{U}\ \mathbf{D}\ \mathbf{U'}$

`d` (numeric vector) Eigenvalues from spectral value decomposition of $\mathbf{G} = \mathbf{U}\ \mathbf{D}\ \mathbf{U'}$

`theta` (numeric) Residual/genetic variances ratio. When it is not `NULL`, the optimization of the likelihood function (REML or ML) is not performed

`BLUP` `TRUE` or `FALSE` to whether return the random effects estimates

`method` (character) Either 'REML' (Restricted Maximum Likelihood) or 'ML' (Maximum Likelihood)

`return.Hinv` `TRUE` or `FALSE` to whether return the inverse of the matrix H

`tol` (numeric) Maximum error between two consecutive solutions (convergence tolerance) when finding the root of the log-likelihood's first derivative

`maxiter` (integer) Maximum number of iterations to run before convergence is reached

`interval` (numeric vector) Range of values in which the root is searched

`warn` `TRUE` or `FALSE` to whether show warnings

Details

The basic linear mixed model that relates phenotypes with genetic values is of the form

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{Z}\mathbf{u} + \mathbf{e}$$

where **y** is a vector with the response, **b** is the vector of fixed effects, **u** is the vector of the (random) genetic values of the genotypes, **e** is the vector of environmental residuals (random error), and **X** and **Z** are design matrices conecting the fixed and genetic effects with replicates. Genetic values are assumed to follow a Normal distribution as $\mathbf{u} \sim N(\mathbf{0}, \sigma^2_u\mathbf{K})$, and the error terms are assumed $\mathbf{e} \sim N(\mathbf{0}, \sigma^2_e\mathbf{D})$, with $\mathbf{D} = \mathbf{I}$ being an identity matrix.

The vector of genetic values $\mathbf{g} = \mathbf{Z}\mathbf{u}$ will therefore follow $\mathbf{g} \sim N(\mathbf{0}, \sigma^2_u\mathbf{G})$ where $\mathbf{G} = \mathbf{Z}\mathbf{K}\mathbf{Z}'$. In the un-replicated case, $\mathbf{Z} = \mathbf{I}$ is an identity matrix, and hence $\mathbf{g} = \mathbf{u}$ and $\mathbf{G} = \mathbf{K}$.

The predicted values $\mathbf{u}_{trn} = (u_i)$, $i = 1, 2, ..., n_{trn}$, corresponding to observed data (training set) are derived as

$$\mathbf{u}_{tst} = \mathbf{H}(\mathbf{y}_{trn} - \mathbf{X}_{trn}\mathbf{b})$$

where **H** is a matrix of weights given by

$$\mathbf{H} = \mathbf{G}_{trn}(\mathbf{G}_{trn} + \theta\mathbf{D})^{-1}$$

where $\mathbf{G}_{trn}$ is the sub-matrix corresponding to the training set, and $\theta = \sigma^2_e/\sigma^2_u$ is the residual/genetic variances ratio representing a shrinkage parameter. This parameter is expressed in terms of the heritability, $h^2 = \sigma^2_u/(\sigma^2_u + \sigma^2_e)$, as $\theta = (1 - h^2)/h^2$.

The predictions of $\mathbf{u}_{tst}$ corresponding to un-observed data (testing set) can be obtained by using

$$\mathbf{H} = \mathbf{G}_{tst,trn}(\mathbf{G}_{trn} + \theta\mathbf{D})^{-1}$$

where $\mathbf{G}_{tst,trn}$ is the sub-matrix of $\mathbf{G}$ corresponding to the testing set (in rows) and training set (in columns).

Solutions are found using the GEMMA (Genome-wide Efficient Mixed Model Analysis) approach (Zhou & Stephens, 2012). First, the Brent's method is implemented to solve for the genetic/residual variances ratio (i.e., $1/\theta$) from the first derivative of the log-likelihood (either REML or ML). Then, variances σ^2_u and σ^2_e are calculated. Finally, $\mathbf{b}$ is obtained using Generalized Least Squares.

Value

Returns a list object that contains the elements:

- `b`: (vector) fixed effects solutions (including the intercept).
- `u`: (vector) random effects solutions.
- `varU`: random effect variance.
- `varE`: residual variance.
- `h2`: heritability.
- `convergence`: (logical) whether Brent's method converged.
- `method`: either 'REML' or 'ML' method used.

Author(s)

Paulino Perez-Rodriguez, Marco Lopez-Cruz (<`maraloc@gmail.com`>) and Gustavo de los Campos

References

VanRaden PM (2008). Efficient methods to compute genomic predictions. *Journal of Dairy Science*, **91**(11), 4414–4423.

Zhou X, Stephens M (2012). Genome-wide efficient mixed-model analysis for association studies. *Nature Genetics*, **44**(7), 821-824

Examples

```
require(SFSI)
data(wheatHTP)

index = which(Y$CV %in% 1:2)       # Use only a subset of data
M = scale(M[index,])/sqrt(ncol(M))   # Subset and scale markers
G = tcrossprod(M)                    # Genomic relationship matrix
y = as.vector(scale(Y[index,"E1"]))  # Subset response variable

# Training and testing sets
tst = seq(1,length(y),by=3)
trn = seq_along(y)[-tst]

yNA <- y
yNA[tst] <- NA
fm1 = fitBLUP(yNA, K=G)
plot(y[tst],fm1$u[tst])      # Predicted vs observed values in testing set
cor(y[tst],fm1$u[tst])       # Prediction accuracy in testing set
cor(y[trn],fm1$u[trn])       # Prediction accuracy in training set
fm1$theta                    # Residual/Genetic variances ratio
fm1$h2                       # Heritability


if(requireNamespace("float")){
 # Using a 'float' type variable
 G2 = float::fl(G)
 fm2 = fitBLUP(yNA, K=G2)
 max(abs(fm1$u-fm2$u))  # Check for discrepances
}
```

---

getGenCov *Genetic covariances*

Description

Pairwise genetic covariances for variables with the same experimental design and equal variance

Usage

```
getGenCov(y1, y2, X = NULL, Z = NULL, K = NULL,
          U = NULL, d = NULL, scale = TRUE,
          mc.cores = 1, warn = FALSE, ...)
```

Arguments

y1	(numeric vector) Response variable 1

`y2` (numeric matrix) Response variable 2. The number of rows must be equal to length of vector y1

`X` (numeric matrix) Design matrix for fixed effects. When `X=NULL` a vector of ones is constructed only for the intercept (default)

`Z` (numeric matrix) Design matrix for random effects. When `Z=NULL` an identity matrix is considered (default) thus **G = K**; otherwise $\mathbf{G} = \mathbf{Z}\ \mathbf{K}\ \mathbf{Z'}$ is used

`K` (numeric matrix) Kinship relationships

`U` (numeric matrix) Eigenvectors from spectral value decomposition of $\mathbf{G} = \mathbf{U}\ \mathbf{D}\ \mathbf{U'}$

`d` (numeric vector) Eigenvalues from spectral value decomposition of $\mathbf{G} = \mathbf{U}\ \mathbf{D}\ \mathbf{U'}$

`scale` `TRUE` or `FALSE` to scale y1 and y2 by their corresponding standard deviations so the resulting variables will have unit variance

`mc.cores` (integer) Number of cores used. The analysis is run in parallel when `mc.cores` is greater than 1. Default is `mc.cores=1`

`warn` `TRUE` or `FALSE` to whether show warnings

`...` Other arguments passed to the function 'fitBLUP'

Details

Assumes that both $\mathbf{y}_1$ and $\mathbf{y}_2$ follow the basic linear mixed model that relates phenotypes with genetic values of the form

$$\mathbf{y}_1 = \mathbf{X}\mathbf{b}_1 + \mathbf{Z}\mathbf{u}_1 + \mathbf{e}_1$$

$$\mathbf{y}_2 = \mathbf{X}\mathbf{b}_2 + \mathbf{Z}\mathbf{u}_2 + \mathbf{e}_2$$

where $\mathbf{b}_1$ and $\mathbf{b}_2$ are the specific fixed effects, $\mathbf{u}_1$ and $\mathbf{u}_2$ are the specific genetic values of the genotypes, $\mathbf{e}_1$ and $\mathbf{e}_2$ are the vectors of specific environmental residuals, and $\mathbf{X}$ and $\mathbf{Z}$ are common design matrices conecting the fixed and genetic effects with replicates. Genetic values are assumed to follow a Normal distribution as $\mathbf{u}_1 \sim N(\mathbf{0}, \sigma^2_{u_1}\mathbf{K})$ and $\mathbf{u}_2 \sim N(\mathbf{0}, \sigma^2_{u_2}\mathbf{K})$, and environmental terms are assumed $\mathbf{e}_1 \sim N(\mathbf{0}, \sigma^2_{e_1}\mathbf{I})$ and $\mathbf{e}_2 \sim N(\mathbf{0}, \sigma^2_{e_2}\mathbf{I})$.

The genetic covariance $\sigma^2_{u_1 u_2}$ is estimated from the formula for the variance for the sum of two variables as

$$\sigma^2_{u_1 u_2} = \frac{1}{2}(\sigma^2_{u_3} - \sigma^2_{u_1} - \sigma^2_{u_2})$$

where $\sigma^2_{u_3}$ is the genetic variance of the variable $\mathbf{y}_3 = \mathbf{y}_1 + \mathbf{y}_2$ that also follows the same model as for $\mathbf{y}_1$ and $\mathbf{y}_2$.

Likewise, the environmental covariance $\sigma^2_{e_1 e_2}$ is estimated as

$$\sigma^2_{e_1 e_2} = \frac{1}{2}(\sigma^2_{e_3} - \sigma^2_{e_1} - \sigma^2_{e_2})$$

where $\sigma^2_{e_3}$ is the error variance of the variable $\mathbf{y}_3$.

Solutions are found using the function 'fitBLUP' (see `help(fitBLUP)`) to sequentialy fit mixed models for all the variables $\mathbf{y}_1$, $\mathbf{y}_2$ and $\mathbf{y}_3$.

Value

Returns a list object that contains the elements:

- `varU1`: genetic variance for response variable 1.
- `varU2`: (vector) genetic variances for response variable 2.
- `varE1`: error variance for response variable 1.
- `varE2`: (vector) error variances for response variable 2.
- `covU`: (vector) genetic covariances between response variables 1 and 2.
- `covE`: (vector) environmental covariances between response variables 1 and 2.

Author(s)

Marco Lopez-Cruz (<`maraloc@gmail.com`>) and Gustavo de los Campos

Examples

```
require(SFSI)
data(wheatHTP)

index = which(Y$CV %in% 1:2)        # Use only a subset of data
M = scale(M[index,])/sqrt(ncol(M))    # Subset and scale markers
G = tcrossprod(M)                     # Genomic relationship matrix
y = as.vector(scale(Y[index,"E1"]))   # Subset response variable
X = scale(X_E1[index,20:50])          # Subset reflectance data

fm = getGenCov(y,X,K=G)

covU = fm$covU                     # Genetic covariance
covP_corrected = fm$covU+fm$covE   # Phenotypic covariance
covP_uncorrected = cov(y,X)        # Sample phenotypic covariance

plot(covP_corrected,covP_uncorrected)
plot(covU,covP_uncorrected)
plot(covU,covP_corrected)
```

LARS *Least Angle Regression to solve LASSO-type problems*

Description

Computes the entire LASSO solution for the regression coefficients, starting from zero, to the least-squares estimates, via the Least Angle Regression (LARS) algorithm (Efron, 2004). It uses as inputs a variance matrix among predictors and a covariance vector between response and predictors.

Usage

```
LARS(Sigma, Gamma, method = c("LAR", "LAR-LASSO"), dfmax = NULL,
  eps = .Machine$double.eps, scale = TRUE, verbose = FALSE)
```

Arguments

`Sigma` (numeric matrix) Variance-covariance matrix of predictors. It can be of the "float32" type as per the ’float’ R-package

`Gamma` (numeric vector) Covariance between response variable and predictors

`method` (character) Either:

- `'LAR'`: Computes the entire sequence of all coefficients. Values of lambdas are calculated at each step.
- `'LAR-LASSO'`: Similar to `'LAR'` but solutions when a predictor leaves the solution are also returned.

Default is `method='LAR'`

`dfmax` (integer) Maximum number of non-zero coefficients in the last LARS solution. Default `dfmax=NULL` will calculate solutions for the entire lambda sequence

`eps` (numeric) An effective zero. Default is the machine precision

`scale` `TRUE` or `FALSE` to scale matrix `Sigma` for variables with unit variance and scale `Gamma` by the standard deviation of the corresponding predictor taken from the diagonal of `Sigma`

`verbose` `TRUE` or `FALSE` to whether printing each LARS step

Details

Finds solutions for the regression coefficients in a linear model

$$y_i = \mathbf{x}_i'\boldsymbol{\beta} + e_i$$

where y_i is the response for the i^{th} observation, $\mathbf{x}_i = (x_{i1}, ..., x_{ip})'$ is a vector of p predictors assumed to have unit variance, $\boldsymbol{\beta} = (\beta_1, ..., \beta_p)'$ is a vector of regression coefficients, and e_i is a residual.

The regression coefficients $\boldsymbol{\beta}$ are estimated as function of the variance matrix among predictors ($\boldsymbol{\Sigma}$) and the covariance vector between response and predictors ($\boldsymbol{\Gamma}$) by minimizing the penalized mean squared error function

$$-\boldsymbol{\Gamma}'\boldsymbol{\beta} + 1/2\boldsymbol{\beta}'\boldsymbol{\Sigma}\boldsymbol{\beta} + 1/2\lambda||\boldsymbol{\beta}||_1$$

where λ is the penalization parameter and $||\boldsymbol{\beta}||_1 = \sum_{j=1}^{p} |\beta_j|$ is the L1-norm.

The algorithm to find solutions for each β_j is fully described in Efron (2004) in which the "current correlation" between the predictor x_{ij} and the residual $e_i = y_i - \mathbf{x}_i'\boldsymbol{\beta}$ is expressed (up-to a constant) as

$$r_j = \Gamma_j - \boldsymbol{\Sigma}_j'\boldsymbol{\beta}$$

where Γ_j is the j^{th} element of $\boldsymbol{\Gamma}$ and $\boldsymbol{\Sigma}_j$ is the j^{th} column of the matrix $\boldsymbol{\Sigma}$

Value

Returns a list object with the following elements:

- `lambda`: (vector) all the sequence of values of the LASSO penalty.
- `beta`: (matrix) regression coefficients for each predictor (in rows) associated to each value of the penalization parameter lambda (in columns).
- `df`: (vector) degrees of freedom, number of non-zero predictors associated to each value of lambda.

The returned object is of the class 'LASSO' for which methods `fitted` exist. Function `path.plot` can be also used

Author(s)

Marco Lopez-Cruz (<`maraloc@gmail.com`>) and Gustavo de los Campos. Adapted from the 'lars' function in package 'lars' (Hastie & Efron, 2013)

References

Efron B, Hastie T, Johnstone I, Tibshirani R (2004). Least angle regression. *The Annals of Statistics*, **32**(2), 407–499.

Friedman J, Hastie T, Tibshirani R(2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, **33**(1), 1–22.

Hastie T, Efron B (2013). lars: least angle regression, Lasso and forward stagewise. https://cran.r-project.org/package=lars.

Tibshirani R (1996). Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society B*, **58**(1), 267–288.

Examples

```
require(SFSI)
data(wheatHTP)

y = as.vector(Y[,"E1"])  # Response variable
X = scale(X_E1)           # Predictors

# Training and testing sets
tst = 1:ceiling(0.3*length(y))
trn = seq_along(y)[-tst]

# Calculate covariances in training set
XtX = var(X[trn,])
Xty = cov(y[trn],X[trn,])

# Run the penalized regression
fm1 = LARS(XtX,Xty,method="LAR-LASSO")

# Predicted values
yHat1 = fitted(fm1, X=X[trn,])  # training data
```

```
yHat2 = fitted(fm1, X=X[tst,])  # testing data

# Penalization vs correlation
plot(-log(fm1$lambda[-1]),cor(y[trn],yHat1[,-1]), main="training")
plot(-log(fm1$lambda[-1]),cor(y[tst],yHat2[,-1]), main="testing")


if(requireNamespace("float")){
 # Using a 'float' type variable
 XtX2 = float::fl(XtX)
 fm2 = LARS(XtX2,Xty,method="LAR-LASSO")
 max(abs(fm1$beta-fm2$beta))      # Check for discrepances in beta
 max(abs(fm1$lambda-fm2$lambda))  # Check for discrepances in lambda
}
```

Methods_LASSO *LASSO methods*

Description

Predicted values for a provided matrix of predictors X

Usage

```
## S3 method for class 'LASSO'
fitted(object, ...)
```

Arguments

object	An object of the class 'LASSO' returned either by the function 'LARS' or 'solveEN'
...	Other arguments: X (numeric matrix) scores for as many predictors there are in ncol(object$beta) (in columns) for a desired number n of observations (in rows)

Value

Returns a matrix that contains, for each value of lambda (in columns), the predicted values corresponding to each row of the matrix X

Author(s)

Marco Lopez-Cruz (<maraloc@gmail.com>) and Gustavo de los Campos

Examples

```
require(SFSI)
data(wheatHTP)

y = as.vector(Y[,”E1”])  # Response variable
X = scale(X_E1)          # Predictors

# Training and testing sets
tst = seq(1,length(y),by=3)
trn = seq_along(y)[-tst]

# Calculate covariances in training set
XtX = var(X[trn,])
Xty = cov(y[trn],X[trn,])

# Run the penalized regression
fm = solveEN(XtX,Xty,alpha=0.5)

# Predicted values
yHat1 = fitted(fm, X=X[trn,])  # training data
yHat2 = fitted(fm, X=X[tst,])  # testing data

# Penalization vs correlation
plot(-log(fm$lambda[-1]),cor(y[trn],yHat1[,-1]), main=”training”)
plot(-log(fm$lambda[-1]),cor(y[tst],yHat2[,-1]), main=”testing”)
```

`Methods_SSI` *SSI methods*

Description

Useful methods for retrieving, summarizing and visualizing important results from an object of the class ’SSI’

Usage

```
## S3 method for class 'SSI'
coef(object, ..., df=NULL, tst=NULL)

## S3 method for class 'SSI'
fitted(object, ...)

## S3 method for class 'SSI'
summary(object, ...)

## S3 method for class 'SSI'
plot(..., py=c(”accuracy”,”MSE”), nbreaks.x=6)
```

Arguments

`object` An object of the class 'SSI'

`...` Arguments to be passed:

- One or more objects of the class 'SSI' (for method `plot`)
- Other arguments for method `plot`: 'xlab', 'ylab', 'main', 'lwd', 'xlim', 'ylim'
- Not needed for methods `summary`, `coef` and `fitted`

`df` (numeric) Average (across testing individuals) number of non-zero regression coefficients

`tst` (integer vector) Which elements from vector `y` (stored in `object$y`) are in testing set. Default `tst=NULL` will consider all individuals in `object$tst`

`py` (character) Either 'accuracy' (correlation between observed and predicted values) or 'MSE' (mean squared error) to plot in the y-axis

`nbreaks.x` (integer) Number of breaks in the x-axis

Value

Method `fitted` returns a matrix with the predicted values for each individual in the testing set (in rows) for each value of lambda (in columns).

Method `coef` (list of matrices) returns the regression coefficients for each testing set individual (elements of the list). Each matrix contains the coefficients for each value of lambda (in rows) associated to each training set individual (in columns). If `tst` is specified, the elements of the list will correspond only to the testing individuals given in `tst`. If `df` is specified, only the coefficients for the lambda associated to `df` are returned as a 'matrix' with testing individuals in rows.

Method `summary` returns a list object containing:

- `lambda`: (vector) sequence of values of lambda used in the coefficients' estimation.
- `df`: (vector) degrees of freedom (across testing individuals) at each solution associated to each value of lambda.
- `accuracy`: (vector) correlation between observed and predicted values associated to each value of lambda.
- `MSE`: (vector) mean squared error associated to each value of lambda.
- `optCOR`: (vector) summary of the SSI with maximum accuracy.
- `optMSE`: (vector) summary of the SSI with minimum MSE.

Method `plot` creates a plot of either accuracy or MSE versus the (average across testing individuals) number of predictors (with non-zero regression coefficient) and versus lambda.

Author(s)

Marco Lopez-Cruz (<`maraloc@gmail.com`>) and Gustavo de los Campos

Examples

```
require(SFSI)
data(wheatHTP)

index = which(Y$CV %in% 1:2)        # Use only a subset of data
M = scale(M[index,])/sqrt(ncol(M))    # Subset and scale markers
G = tcrossprod(M)                     # Genomic relationship matrix
y = as.vector(scale(Y[index,"E1"]))   # Subset ans scale response variable

# Training and testing sets
tst = seq(1,length(y),by=5)
trn = seq_along(y)[-tst][1:200]

fm1 = SSI(y,K=G,tst=tst,trn=trn)

yHat = fitted(fm1)                # Predicted values for each SSI
out = summary(fm1)                # Useful function to get results
corTST = out$accuracy             # Testing set accuracy (correlation cor(y,yHat))
out$optCOR                        # SSI with maximum accuracy
out$optMSE                        # SSI with minimum MSE
B = coef(fm1)                     # Regression coefficients
B = coef(fm1,df=out$optCOR['df']) # Regression coefficients associated with one 'df'
plot(fm1,main=expression('corr('*y[obs]*','*y[pred]*') vs sparsity'))
plot(fm1,py="MSE",ylab='Mean Square Error', xlab='Sparsity')
```

net.plot *Network plot*

Description

Network plot of testing and training individuals from an object of the class ’SSI’

Usage

```
net.plot(object, Z = NULL, K = NULL, tst = NULL,
        show.names = FALSE, group = NULL, group.shape = NULL,
        set.color = NULL, set.size = NULL, df = NULL, main,
        axis.labels = TRUE, curve = FALSE, bg.color = "white",
        unified = TRUE, ntst = 36, line.color = "gray80",
        line.tick = 0.3, legend.pos="right", point.color = "gray20",
        sets = c("Testing","Supporting","Non-active"),
        eps = .Machine$double.eps)
```

Arguments

`object` An object of the ’SSI’ class or a matrix of coefficients

`Z` (numeric matrix) Design matrix for random effects. When `Z=NULL` an identity matrix is considered (default) thus **G = K**; otherwise **G** = **Z K Z’** is used

`K` (numeric matrix) Kinship relationships. This can be a (character) name of a binary file where the matrix is stored

`tst` (integer vector) Which elements from vector `y` (stored in `fm$y`) are in testing set and to plot. They must be contained in `fm$tst`. Default `tst=NULL` will consider the whole vector `fm$tst` to plot

`group` (data.frame) Column grouping for the individuals. The rows must match with the rows in `G` matrix

`df` (numeric) Average number of training individuals contributing to the prediction (active) of testing individuals. Default `df=NULL` will use the `df` that yielded the optimal accuracy

`main` (character/expression) Title of the plot

`bg.color` (character) Plot background color

`line.color` (character) Color of lines connecting nodes in rows with those in columns

`line.tick` (numeric) Tick of lines connecting nodes in rows with those in columns

`curve` `TRUE` or `FALSE` to whether draw curve lines connecting nodes in rows with those in columns

`show.names` `TRUE` or `FALSE` to whether show node names given by the row/column names of either `K` or `object` (when this is a matrix)

`set.color` (character vector) Color point of each type of node: row, ’active’ column, and ’non-active’ column, respectively

`set.size` (numeric vector) Size of each type of node: row, ’active’ column, and ’non-active’ column, respectively

`group.shape` (integer vector) Shape of each level of the grouping column provided as `group`

`axis.labels` `TRUE` or `FALSE` to whether show labels in both axes

`unified` `TRUE` or `FALSE` to whether show an unified plot or separated for each individual in ’testing’

`point.color` (character) Color of the points in the plot

`ntst` (integer) Maximum number of row nodes (’testing’) that are plotted separated as indicated by `unified=FALSE`

`legend.pos` (character) Either "right", topright","bottomleft","bottomright","topleft", or "none" indicating where the legend is positioned in the plot

`sets` (character vector) Names of the types of node: row, ’active’ column, and ’non-active’ column, respectively

`eps` (numeric) Minumum value to declare nodes to be connected. Default is the machine precision (numerical zero)

Details

Plot edges in the plane xy given by a numerical matrix. When the object is a ’SSI’ object the edges are taken from the regression coefficients from the ’SSI’. Edges are plotted according to the Fruchterman-Reingold algorithm. When a matrix **K** is provided, edges are plotted according to the spectral value decomposition of **Z K Z’** = **U D U’**

Value

Returns the top-2 PC's plot connecting testing (predicted) individuals with training (predictors) individuals

Author(s)

Marco Lopez-Cruz (<maraloc@gmail.com>) and Gustavo de los Campos

Examples

```
require(SFSI)
data(wheatHTP)

index = which(Y$CV == 1)             # Use only a subset of data
M = scale(M[index,])/sqrt(ncol(M))   # Subset and scale markers
G = tcrossprod(M)                    # Genomic relationship matrix
y = as.vector(scale(Y[index,"E1"]))  # Subset ans scale response variable

# Training and testing sets
tst = seq(20,length(y),by=10)
trn = seq_along(y)[-tst]

fm = SSI(y,K=G,tst=tst,trn=trn)

# Basic setting
net.plot(fm)
net.plot(fm, show.names=c(TRUE,TRUE,FALSE), set.size=c(4,2,1), df=10)


# Passing a matrix of coefficients
B = as.matrix(coef(fm,df=15))
net.plot(B, curve=TRUE, set.size=c(3.5,1.5,1))
```

path.plot *Coefficients path plot*

Description

Coefficients evolution path plot from an object of the class 'LASSO' or 'SSI'

Usage

```
path.plot(object, Z = NULL, K = NULL, tst = NULL,
          cor.max = 0.85, nbreaks.x=6, ...)
```

Arguments

`object` An object of the 'LASSO' or 'SSI' class

`Z` (numeric matrix) Design matrix for random effects. When `Z=NULL` an identity matrix is considered (default) thus **G = K**; otherwise $\mathbf{G} = \mathbf{Z}\ \mathbf{K}\ \mathbf{Z'}$ is used. Only needed for a `fm` object of the class 'SSI'

`K` (numeric matrix) Kinship relationships. This can be a name of a binary file where the matrix is stored. Only needed for a `fm` object of the class 'SSI'

`tst` (integer vector) Which elements from vector `y` (stored in `fm$y`) are in testing set and to plot. They must be contained in `fm$tst`. Default `tst=NULL` will consider the whole vector `fm$tst` to plot

`cor.max` (numeric) Maximum correlation allowed for two different coefficients. A group of coeffcients with a correlation greater than `cor.max` are likely to overlap in the plot thus only one is kept

`nbreaks.x` (integer) Number of breaks in the x-axis

`...` Other arguments for method `plot`: 'xlab', 'ylab', 'main', 'lwd'

Value

Returns the plot of the coefficients' evolution path along the regularization parameter

Author(s)

Marco Lopez-Cruz (<`maraloc@gmail.com`>) and Gustavo de los Campos

Examples

```
require(SFSI)
data(wheatHTP)

index = which(Y$CV == 1)             # Use only a subset of data
M = scale(M[index,])/sqrt(ncol(M))   # Subset and scale markers
G = tcrossprod(M)                    # Genomic relationship matrix
y = as.vector(scale(Y[index,"E1"]))  # Subset response variable
X = scale(X_E1[index,])              # Reflectance data

# Sparse phenotypic regression
fm1 = LARS(var(X),cov(y,X))

# Sparse family index
fm2 = SSI(y,K=G,tst=1:10,trn=11:50)


path.plot(fm1)
path.plot(fm2, cor.max=0.6)
path.plot(fm2, K=G, cor.max=0.6)


# Path plot for the first individual in testing set for the SSI
path.plot(fm2, K=G, tst=fm2$tst[1])
```

solveEN *Coordinate Descent algorithm to solve Elastic-Net-type problems*

Description

Computes the entire Elastic-Net solution for the regression coefficients for all values of the penalization parameter, via the Coordinate Descent (CD) algorithm (Friedman, 2007). It uses as inputs a variance matrix among predictors and a covariance vector between response and predictors

Usage

```
solveEN(Sigma, Gamma, alpha = 1, lambda = NULL,
        nlambda = 100, lambda.min = .Machine$double.eps^0.5,
        dfmax = NULL, scale = TRUE, tol = 1e-05,
        maxiter = 1000, verbose = FALSE)
```

Arguments

Sigma (numeric matrix) Variance-covariance matrix of predictors. It can be of the "float32" type as per the 'float' R-package

Gamma (numeric vector) Covariance between response variable and predictors

lambda (numeric vector) Penalization parameter sequence. Default is `lambda=NULL`, in this case a decreasing grid of `'nlambda'` lambdas will be generated starting from a maximum equal to

```
max(abs(Gamma)/alpha)
```

to a minimum equal to zero. If `alpha=0` the grid is generated starting from a maximum equal to 5

nlambda (integer) Number of lambdas generated when `lambda=NULL`

lambda.min (numeric) Minimum value of lambda that are generated when `lambda=NULL`

alpha (numeric) Value between 0 and 1 for the weights given to the L1 and L2-penalties

scale `TRUE` or `FALSE` to scale matrix `Sigma` for variables with unit variance and scale `Gamma` by the standard deviation of the corresponding predictor taken from the diagonal of `Sigma`

tol (numeric) Maximum error between two consecutive solutions of the CD algorithm to declare convergence

maxiter (integer) Maximum number of iterations to run the CD algorithm at each lambda step before convergence is reached

dfmax (integer) Maximum number of non-zero coefficients in the last solution. Default `dfmax=NULL` will calculate solutions for the entire lambda grid

verbose `TRUE` or `FALSE` to whether printing each CD step

Details

Finds solutions for the regression coefficients in a linear model

$$y_i = \mathbf{x}_i'\boldsymbol{\beta} + e_i$$

where y_i is the response for the i^{th} observation, $\mathbf{x}_i = (x_{i1}, ..., x_{ip})'$ is a vector of p predictors assumed to have unit variance, $\boldsymbol{\beta} = (\beta_1, ..., \beta_p)'$ is a vector of regression coefficients, and e_i is a residual.

The regression coefficients $\boldsymbol{\beta}$ are estimated as function of the variance matrix among predictors ($\boldsymbol{\Sigma}$) and the covariance vector between response and predictors ($\boldsymbol{\Gamma}$) by minimizing the penalized mean squared error function

$$-\boldsymbol{\Gamma}'\boldsymbol{\beta} + 1/2\boldsymbol{\beta}'\boldsymbol{\Sigma}\boldsymbol{\beta} + \lambda J(\boldsymbol{\beta})$$

where λ is the penalization parameter and $J(\boldsymbol{\beta})$ is a penalty function given by

$$1/2(1-\alpha)||\boldsymbol{\beta}||_2^2 + \alpha||\boldsymbol{\beta}||_1$$

where $0 \leq \alpha \leq 1$, and $||\boldsymbol{\beta}||_1 = \sum_{j=1}^p |\beta_j|$ and $||\boldsymbol{\beta}||_2^2 = \sum_{j=1}^p \beta_j^2$ are the L1 and (squared) L2-norms, respectively.

The "partial residual" excluding the contribution of the predictor x_{ij} is

$$e_i^{(j)} = y_i - \mathbf{x}_i'\boldsymbol{\beta} + x_{ij}\beta_j$$

then the ordinary least-squares (OLS) coefficient of x_{ij} on this residual is (up-to a constant)

$$\beta_j^{(ols)} = \Gamma_j - \boldsymbol{\Sigma}_j'\boldsymbol{\beta} + \beta_j$$

where Γ_j is the j^{th} element of $\boldsymbol{\Gamma}$ and $\boldsymbol{\Sigma}_j$ is the j^{th} column of the matrix $\boldsymbol{\Sigma}$.

Coefficients are updated for each $j = 1, ..., p$ from their current value β_j to a new value $\beta_j(\alpha, \lambda)$, given α and λ, by "soft-thresholding" their OLS estimate until convergence as fully described in Friedman (2007).

Value

Returns a list object containing the elements:

- `lambda`: (vector) all the sequence of values of the penalty.
- `beta`: (matrix) regression coefficients for each predictor (in rows) associated to each value of the penalization parameter lambda (in columns).
- `df`: (vector) degrees of freedom, number of non-zero predictors associated to each value of lambda.

The returned object is of the class 'LASSO' for which methods `fitted` exist. Function 'path.plot' can be also used

Author(s)

Marco Lopez-Cruz (<maraloc@gmail.com>) and Gustavo de los Campos

References

Friedman J, Hastie T, Höfling H, Tibshirani R (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics*, **1**(2), 302–332.

Hoerl AE, Kennard RW (1970). Ridge Regression: Biased estimation for nonorthogonal problems. *Technometrics*, **12**(1), 55–67.

Tibshirani R (1996). Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society B*, **58**(1), 267–288.

Zou H, Hastie T (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B*, **67**(2), 301–320.

Examples

```
require(SFSI)
data(wheatHTP)

y = as.vector(Y[,"E1"])  # Response variable
X = scale(X_E1)          # Predictors

# Training and testing sets
tst = seq(1,length(y),by=3)
trn = seq_along(y)[-tst]

# Calculate covariances in training set
XtX = var(X[trn,])
Xty = cov(y[trn],X[trn,])

# Run the penalized regression
fm1 = solveEN(XtX,Xty,alpha=0.5)

# Predicted values
yHat1 = fitted(fm1, X=X[trn,])  # training data
yHat2 = fitted(fm1, X=X[tst,])  # testing data

# Penalization vs correlation
plot(-log(fm1$lambda[-1]),cor(y[trn],yHat1[,-1]), main="training")
plot(-log(fm1$lambda[-1]),cor(y[tst],yHat2[,-1]), main="testing")


if(requireNamespace("float")){
 # Using a 'float' type variable
 XtX2 = float::fl(XtX)
 fm2 = solveEN(XtX2,Xty,alpha=0.5)
 max(abs(fm1$beta-fm2$beta))  # Check for discrepances
}
```

SSI *Sparse Selection Index*

Description

Computes the entire Elastic-Net solution for the regression coefficients of a Selection Index for a grid of values of the penalization parameter.

An optimal penalization can be chosen using cross-validation (CV) within a specific training set.

Usage

```
SSI(y, X = NULL, b = NULL, Z = NULL, K, D = NULL,
    theta = NULL, h2 = NULL, trn = seq_along(y), tst = seq_along(y),
    subset = NULL, alpha = 1, lambda = NULL, nlambda = 100,
    lambda.min = .Machine$double.eps^0.5, common.lambda = TRUE,
    tol = 1E-4, maxiter = 500, method = c("REML","ML"),
    save.at = NULL, name = NULL, mc.cores = 1, verbose = TRUE)

SSI.CV(y, X = NULL, b = NULL, Z = NULL, K, D = NULL,
      theta = NULL, h2 = NULL, trn = seq_along(y), alpha = 1,
      lambda = NULL, nlambda = 100, lambda.min = .Machine$double.eps^0.5,
      nCV = 1, nfolds = 5, seed = NULL, common.lambda = TRUE,
      tol = 1E-4, maxiter = 500, method = c("REML","ML"),
      name = NULL, mc.cores = 1, verbose = TRUE)
```

Arguments

y	(numeric vector) Response variable
X	(numeric matrix) Design matrix for fixed effects. When X=NULL, a vector of ones is constructed only for the intercept (default)
b	(numeric vector) Fixed effects. When b=NULL, only the intercept is estimated from training data using generalized least squares (default)
Z	(numeric matrix) Design matrix for random effects. When Z=NULL an identity matrix is considered (default) thus **G = K**; otherwise **G** = **Z K Z'** is used
K	(numeric matrix) Kinship relationships. This can be of the "float32" type as per the 'float' R-package, or a (character) name of a binary file where the matrix is stored
D	(numeric matrix) Relationships among residuals (usually an identity matrix). When D=NULL an identity matrix is considered (default)
theta	(numeric) Residual/genetic variances ratio. When theta=NULL (default), it is calculated from training data using the function 'fitBLUP' (see help(fitBLUP))
h2	(numeric) Heritability of the response variable. When h2=NULL (default), it is calculated from training data using the function 'fitBLUP' (see help(fitBLUP)). It is used to set the residual/genetic variances ratio theta

`trn` (integer vector) Which elements from vector `y` are in training set. Default `trn=seq_along(y)` will consider all individuals as training

`tst` (integer vector) Which elements from vector `y` are in testing set (prediction set). Default `tst=seq_along(y)` will consider all individuals as testing

`subset` (integer vector) $c(m, M)$ to fit the model only for the m^{th} subset out of M subsets that the testing set will be divided into. Results can be automatically saved when `saveAt` argument is provided and can be retrieved later using function 'collect' (see `help(collect)`). Default is `subset=NULL` for no subsetting, then the model is fitted for all testing data

`alpha` (numeric) Value between 0 and 1 for the weights given to the L1 and L2-penalties

`lambda` (numeric vector) Penalization parameter sequence. Default is `lambda=NULL`, in this case a decreasing grid of `nlambda` lambdas will be generated starting from a maximum equal to

```
max(abs(G[trn,tst])/alpha)
```

to a minimum equal to zero. If `alpha=0` the grid is generated starting from a maximum equal to 5

`nlambda` (integer) Number of lambdas generated when `lambda=NULL`

`lambda.min` (numeric) Minimum value of lambda in the generated grid when `lambda=NULL`

`nfolds` (integer/character) Either 2,3,5,10 or 'n' indicating the number of non-overlaping folds in which the data is split into to do cross-validation. When `nfolds='n'` leave-one-out CV is performed

`seed` (numeric vector) Seed to fix randomization when creating folds for cross-validation. If it is a vector, a number equal to its length of CV repetitions are performed

`nCV` (integer) Number of CV repetitions to be performed. Default is `nCV=1`

`common.lambda` `TRUE` or `FALSE` to whether computing the coefficients for a grid of lambdas common to all individuals in testing set or for a grid of lambdas specific to each individual in testing set. Default is `common.lambda=TRUE`

`mc.cores` (integer) Number of cores used. The analysis is run in parallel when `mc.cores` is greater than 1. Default is `mc.cores=1`

`tol` (numeric) Maximum error between two consecutive solutions of the CD algorithm to declare convergence

`maxiter` (integer) Maximum number of iterations to run the CD algorithm at each lambda step before convergence is reached

`save.at` (character) Prefix name that will be added to the output files name to be saved, this may include a path. Regression coefficients are saved as a binary file (single-precision: 32 bits, 7 significant digits). Default `save.at=NULL` will no save any output

`method` (character) Either 'REML' (Restricted Maximum Likelihood) or 'ML' (Maximum Likelihood) to calculate variance components as per the function 'fitBLUP'

`name` (character) Name given to the output for tagging purposes. Default `name=NULL` will give the name of the method used

`verbose` `TRUE` or `FALSE` to whether printing each step

Details

The basic linear mixed model that relates phenotypes with genetic values is of the form

$$\mathbf{y} = \mathbf{Xb} + \mathbf{Zg} + \mathbf{e}$$

where $\mathbf{y}$ is a vector with the response, $\mathbf{b}$ is the vector of fixed effects, $\mathbf{g}$ is the vector of the genetic values of the genotypes, $\mathbf{e}$ is the vector of environmental residuals, and $\mathbf{X}$ and $\mathbf{Z}$ are design matrices conecting the fixed and genetic effects with replicates. Genetic values are assumed to follow a Normal distribution as $\mathbf{g} \sim N(\mathbf{0}, \sigma_u^2\mathbf{K})$, and environmental terms are assumed $\mathbf{e} \sim N(\mathbf{0}, \sigma_e^2\mathbf{D})$, usually $\mathbf{D} = \mathbf{I}$.

The resulting vector of genetic values $\mathbf{u} = \mathbf{Zg}$ will therefore follow $\mathbf{u} \sim N(\mathbf{0}, \sigma_u^2\mathbf{G})$ where $\mathbf{G} = \mathbf{ZKZ}'$. In the un-replicated case, $\mathbf{Z} = \mathbf{I}$ is an identity matrix, and hence $\mathbf{u} = \mathbf{g}$ and $\mathbf{G} = \mathbf{K}$.

The values $\mathbf{u}_{tst} = (u_i)$, $i = 1, 2, ..., n_{tst}$, for a testing set are estimated individual-wise using (as predictors) all available observations in a training set as

$$u_i = \boldsymbol{\beta}_i'(\mathbf{y}_{trn} - \mathbf{X}_{trn}\mathbf{b})$$

where $\boldsymbol{\beta}_i$ is a vector of weights that are found separately for each individual in the testing set, by minimizing the penalized mean squared error function

$$-\mathbf{G}_{trn,tst(i)}'\boldsymbol{\beta}_i + 1/2\boldsymbol{\beta}_i'(\mathbf{G}_{trn} + \theta\mathbf{D})\boldsymbol{\beta}_i + \lambda J(\boldsymbol{\beta}_i)$$

where $\mathbf{G}_{trn,tst(i)}$ is the i^{th} column of the sub-matrix of $\mathbf{G}$ whose rows correspond to the training set and columns to the testing set; $\mathbf{G}_{trn}$ is the sub-matrix corresponding to the training set; $\theta = \sigma_e^2/\sigma_u^2$ is the residual to genetic variance ratio that can be expressed in terms of the heritability, $h^2 = \sigma_u^2/(\sigma_u^2 + \sigma_e^2)$, as $\theta = (1 - h^2)/h^2$; λ is the penalization parameter; and $J(\boldsymbol{\beta}_i)$ is a penalty function given by

$$1/2(1 - \alpha)||\boldsymbol{\beta}_i||_2^2 + \alpha||\boldsymbol{\beta}_i||_1$$

where $0 \leq \alpha \leq 1$, and $||\boldsymbol{\beta}_i||_1 = \sum_{j=1}^{n_{trn}} |\beta_{ij}|$ and $||\boldsymbol{\beta}_i||_2^2 = \sum_{j=1}^{n_{trn}} \beta_{ij}^2$ are the L1 and (squared) L2-norms, respectively.

Function 'SSI' calculates each individual solution using the function 'solveEN' (via the Coordinate Descent algorithm, see `help(solveEN)`) by setting the argument `Sigma` equal to $\mathbf{G}_{trn} + \theta\mathbf{D}$ and `Gamma` equal to $\mathbf{G}_{trn,tst(i)}$.

Function 'SSI.CV' performs cross-validation within the training data specified in argument `trn`. Training data is divided into k folds and the SSI is sequentially calculated for (all individuals in) one fold (as testing set) using information from the remaining folds (as training set).

Value

Function 'SSI' returns a list object of the class 'SSI' for which methods `coef`, `fitted`, `plot`, and `summary` exist. Functions 'net.plot' and 'path.plot' can be also used. It contains the elements:

- `b`: (vector) fixed effects solutions (including the intercept).

- `Xb`: (vector) product of the design matrix 'X' times the fixed effects solutions.
- `varU`, `varE`, `h2`: variance components solutions.
- `alpha`: value for the elastic-net weights used.
- `lambda`: (matrix) sequence of values of lambda used (in columns) for each testing individual (in rows).
- `df`: (matrix) degrees of freedom (number of non-zero predictors) at each solution given by lambda for each testing individual (in rows).
- `file_beta`: path where regression coefficients are saved.

Function 'SSI.CV' returns a list object of length `nCV` of the class 'SSI.CV' for which methods `plot` and `summary` exist. Each element is also a list containing the elements:

- `b`: (vector) solutions for the fixed effects (including the intercept) for each fold.
- `varU`, `varE`, `h2`: variance components estimated within each fold.
- `folds`: (matrix) assignation of training individuals to folds used for the cross-validation.
- `accuracy`: (matrix) correlation between observed and predicted values (in testing set) within each fold (in rows).
- `MSE`: (matrix) mean squared error of prediction (in testing set) within each fold (in rows).
- `lambda`: (matrix) with the sequence of values of lambda used (averaged across individuals) within each fold (in rows).
- `df`: (matrix) with the degrees of freedom (averaged across individuals) within each fold (in rows).

Author(s)

Marco Lopez-Cruz (<`maraloc@gmail.com`>) and Gustavo de los Campos

References

Efron B, Hastie T, Johnstone I, Tibshirani R (2004). Least angle regression. *The Annals of Statistics*, **32**(2), 407–499.

Friedman J, Hastie T, Höfling H, Tibshirani R (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics*, **1**(2), 302–332.

Hoerl AE, Kennard RW (1970). Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, **12**(1), 55–67.

Lush JL (1947). Family merit an individual merit as bases for selection. Part I. *The American Naturalist*, **81**(799), 241–261.

Tibshirani R (1996). Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society B*, **58**(1), 267–288.

VanRaden PM (2008). Efficient methods to compute genomic predictions. *Journal of Dairy Science*, **91**(11), 4414–4423.

Zou H, Hastie T (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B*, **67**(2), 301–320

Examples

```
require(SFSI)
data(wheatHTP)

index = which(Y$CV == 1)              # Use only a subset of data
M = scale(M[index,])/sqrt(ncol(M))    # Subset and scale markers
G = tcrossprod(M)                     # Genomic relationship matrix
y = as.vector(scale(Y[index,”E1”]))   # Subset ans scale response variable

# Training and testing sets
tst = seq(1,length(y),by=3)
trn = (seq_along(y))[-tst]

# Calculate variance components ratio using training data
yNA <- y
yNA[tst] <- NA
fm0 = fitBLUP(yNA,K=G)
theta = fm0$varE/fm0$varU
h2 = fm0$varU/(fm0$varU + fm0$varE)

# Sparse selection index
fm1 = SSI(y,K=G,theta=theta,trn=trn,tst=tst)
summary(fm1)$optCOR


if(requireNamespace(”float”)){
# Using a 'float' type variable for K
G2 = float::fl(G)
fm2 = SSI(y,K=G2,theta=theta,trn=trn,tst=tst)
summary(fm2)$optCOR  # compare with above results
}


#---------------------------------------------------
# Predicting a testing set using a value of lambda
# obtained from cross-validation in a traning set
#---------------------------------------------------
# Run a cross validation in training set
fm2 = SSI.CV(y,K=G,theta=theta,trn=trn,nfolds=5,name=”1 5CV”)
lambda = summary(fm2)$optCOR[”lambda”]

# Fit the index with the obtained lambda
fm3 = SSI(y,K=G,theta=theta,trn=trn,tst=tst,lambda=lambda)
summary(fm3)$accuracy        # Testing set accuracy

# Compare the accuracy with that of the non-sparse index (G-BLUP)
cor(fm0$u[tst],y[tst])


# Obtain an 'optimal' lambda by repeating the CV several times
fm22 = SSI.CV(y,K=G,theta=theta,trn=trn,nCV=5,name=”5 5CV”)
plot(fm22,fm2)
```

wheat *Wheat dataset*

Description

The dataset consists of 1,092 inbred wheat lines grouped into 39 trials and grown during the 2013-2014 season at the Norman Borlaug experimental research station in Ciudad Obregon, Sonora, Mexico. Each trial consisted of 28 breeding lines that were arranged in an alpha-lattice design with three replicates and six sub-blocks. The trials were grown in four different environments:

- E1: Flat-Drought (sowing in flat with irrigation of 180 mm through drip system)
- E2: Bed-2IR (sowing in bed with 2 irrigations approximately 250 mm)
- E3: Bed-5IR (bed sowing with 5 normal irrigations)
- E4: Bed-EHeat (bed sowing 30 days before optimal planting date with 5 normal irrigations approximately 500 mm)

1. Phenotypic data. Measurements of grain yield (YLD) were reported as the total plot yield after maturity. Records for YLD are reported as adjusted means from which trial, replicate and sub-block effects were removed. Measurements for days to heading (DTH), days to maturity (DTM), and plant height (PH) were recorded only in the first replicate at each trial and thus no phenotype adjustment was made.

2. Reflectance data. Reflectance data was collected from the fields using both infrared and hyper-spectral cameras mounted on an aircraft on 9 different dates (time-points) between January 10 and March 27th, 2014. During each flight, data from 250 wavelengths ranging from 392 to 850 nm were collected for each pixel in the pictures. The average reflectance of all the pixels for each wavelength was calculated from each of the geo-referenced trial plots and reported as each line reflectance. Data for reflectance and Green NDVI and Red NDVI are reported as adjusted phenotypes from which trial, replicate and sub-block effects were removed. Each data-point matches to each data-point in phenotypic data.

3. Marker data. Lines were sequenced for GBS at 192-plexing on Illumina HiSeq2000 or HiSeq2500 with 1 x 100 bp reads. SNPs were called across all lines anchored to the genome assembly of Chinese Spring (International Wheat Genome Sequencing Consortium 2014). Next, SNP were extracted and filtered so that lines >50% missing data were removed. Markers were recoded as –1, 0, and 1, corresponding to homozygous for the minor allele, heterozygous, and homozygous for the major allele, respectively. Next, markers with a minor allele frequency <0.05 and >15% of missing data were removed. Remaining SNPs with missing values were imputed using the mean of the observed marker genotypes at a given locus.

Replicated and Un-replicated data. The CRAN version includes the `wheatHTP` dataset containing (un-replicated) YLD from all environments E1,...,E4, and reflectance (latest time-point only) data from the environment E1 only. Marker data is also included in the dataset. The phenotypic and reflectance data are averages (line effects from mixed models) for 776 lines evaluated in 28 trials (with at least 26 lines each) for which marker information on 3,438 SNPs is available.

The GitHub (development) version of the SFSI R-package (https://github.com/MarcooLopez/SFSI) includes also the wheatHTP dataset plus wheatHTP.E1,...,wheatHTP.E4 datasets containing replicated (adjusted) phenotypic and reflectance data for all four environments, respectively.

Cross-validation partitions. One random partition of 4-folds is provided for the un-replicated data (wheatHTP dataset) containing 776 individuals and 28 trials. Data from 7 entire trials (25% of 28 the trials) were arbitrarily assigned to each of the 4 folds. The partition consist of an array of length 776 with indices 1, 2, 3, and 4 denoting the fold.

Genetic covariances. Multi-variate Gaussian mixed models were fitted to phenotypes from the un-replicated data (wheatHTP dataset) containing 776 individuals. Bi-variate models were fitted to YLD with each of the 250 wavelengths from environment E1. Tetra-variate models were fitted for YLD from all environments. All models were fitted within each fold (provided partition) using scaled (null mean and unit variance) phenotypes from the remaining 3 folds as training data. Bayesian models were implemented using the 'Multitrait' function from the BGLR R-package with 30,000 iterations discarding 5,000 runs for burning. A marker-derived relationships matrix as in VanRaden (2008) was used to model between-individuals genetic effects; both between-traits genetic and residual covariances were assumed unstructured.

Genetic and residual covariances between YLD and each wavelength (environment E1) are storaged in a matrix of 250 rows and 4 columns (folds). Genetic and residual covariances matrices among YLD within each environment are storaged in a list with 4 elements (folds).

Usage

```
data(wheatHTP)
# data(wheatHTP.E1) # GitHub version
# data(wheatHTP.E2) # GitHub version
# data(wheatHTP.E3) # GitHub version
# data(wheatHTP.E4) # GitHub version
```

Format

Replicated data (wheatHTP.E1,...,wheatHTP.E4 datasets):

- Y: (matrix) phenotypic data for YLD, DTH, DTM, and PH; and the trial in which each genotype was tested.
- X: (9-dimensional list) reflectance data for time-points 1,2,...,9.
- VI: (9-dimensional list) green and red NDVI for time-points 1,2,...,9.

Un-replicted data (wheatHTP dataset):

- Y: (matrix) phenotypic data for YLD in environments E1, E2, E3, and E4; and columns 'trial' and 'CV' (indicating the 4-folds partition).
- M: (matrix) marker data with SNPs in columns.
- X_E1: (matrix) reflectance data for time-point 9 in environment E1.
- genCOV_xy: (matrix) genetic covariances between YLD and each reflectance trait, for each fold (in columns).
- resCOV_xy: (matrix) residual covariances between YLD and each reflectance trait, for each fold (in columns).

- genCOV_yy: (4-dimensional list) genetic covariances matrices for YLD among environments, for each fold.
- resCOV_yy: (4-dimensional list) residual covariances matrices for YLD among environments, for each fold.

Source

International Maize and Wheat Improvement Center (CIMMYT), Mexico.

References

Perez-Rodriguez P, de los Campos G (2014). Genome-wide regression and prediction with the BGLR statistical package. *Genetics*, **198**, 483–495.

VanRaden PM (2008). Efficient methods to compute genomic predictions. *Journal of Dairy Science*, **91**(11), 4414–4423.

Index