Package 'statr'

June 20, 2017

Type Package

Title Matt Galloway Personal R Package

Version 0.1.0

Description This is a personal R package. It contains a number of various R functions for organization and convenience purposes.

URL https://github.com/MGallow/statr

BugReports https://github.com/MGallow/statr/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Imports Rcpp (>= 0.12.10),
RcppArmadillo,
dplyr,
MASS,
alr4,
formatR,
devtools,
RcppParallel,
gmodels

LinkingTo Rcpp,
RcppArmadillo,
RcppParallel

Suggests testthat

SystemRequirements GNU make

R topics documented:

`bsearch` *Bisection search*

Description

Minimizes a univariate strictly pseudoconvex function over the interval [a, b]. This is augmented code from Adam Rothman's STAT 8054 course (2017).

Usage

```
bsearch(dg, a, b, L = 1e-07, quiet = FALSE)
```

Arguments

`dg`	the derivative of the function to minimize, where dg(u, ...) is the function evaluated at u.
`a`	left endpoint of the initial interval of uncertainty.
`b`	right endpoint of the initial interval of uncertainty.
`L`	the maximum length of the final interval of uncertainty.
`quiet`	should the function stay quiet?
`...`	additional argument specifications for dg

Value

returns the midpoint of the final interval of uncertainty.

Examples

```
bsearch(dg, -10, 10, quiet = T)
```

`CV_linearc` *CV Linearc (c++)*

Description

Computes the coefficient estimates for linear regression. ridge regularization and bridge regularization optional. This function is to be used with the 'linearc' function

Usage

```
CV_linearc(X, y, lam = 0L, alpha = 0L, penalty = "none", weights = 0L,
  intercept = TRUE, kernel = FALSE, method = "SVD", tol = 1e-05,
  maxit = 10000, vec = 0L, init = 0L, K = 5L)
```

Arguments

`X`	matrix
`y`	matrix or vector of response values 0,1
`lam`	vector of tuning parameters for ridge regularization term. Defaults to 'lam = 0'
`alpha`	vector of tuning parameters for bridge regularization term. Defaults to 'alpha = 1.5'
`penalty`	choose from c('none', 'ridge', 'bridge'). Defaults to 'none'
`intercept`	Defaults to TRUE
`method`	optimization algorithm. Choose from 'IRLS' or 'MM'. Defaults to 'IRLS'
`tol`	tolerance - used to determine algorithm convergence. Defaults to 1e-5
`maxit`	maximum iterations. Defaults to 1e5
`vec`	optional vector to specify which coefficients will be penalized
`init`	optional initialization for MM algorithm
`K`	specify number of folds in cross validation, if necessary

Value

returns best lambda, best alpha, cv.errors

Examples

```
CV_linearc(X, y, lam = seq(0.1, 2, 0.1), alpha = seq(1.1, 1.9, 0.1), penalty = 'bridge', vec = c(0,1,1,1))
```

CV_logisticc *CV Logisticc (c++)*

Description

Computes the coefficient estimates for logistic regression. ridge regularization and bridge regularization optional. This function is to be used with the 'logisticc' function.

Usage

```
CV_logisticc(X, y, lam = 0L, alpha = 0L, penalty = "none",
  intercept = TRUE, method = "IRLS", tol = 1e-05, maxit = 10000,
  vec = 0L, init = 0L, criteria = "logloss", K = 5L)
```

Arguments

X	matrix
y	matrix or vector of response values 0,1
lam	vector of tuning parameters for ridge regularization term. Defaults to 'lam = 0'
alpha	vector of tuning parameters for bridge regularization term. Defaults to 'alpha = 1.5'
penalty	choose from c('none', 'ridge', 'bridge'). Defaults to 'none'
intercept	Defaults to TRUE
method	optimization algorithm. Choose from 'IRLS' or 'MM'. Defaults to 'IRLS'
tol	tolerance - used to determine algorithm convergence. Defaults to 1e-5
maxit	maximum iterations. Defaults to 1e5
vec	optional vector to specify which coefficients will be penalized
init	optional initialization for MM algorithm
criteria	specify the criteria for cross validation. Choose from c('mse', 'logloss', 'misclass'). Defauls to 'logloss'
K	specify number of folds in cross validation, if necessary

Value

returns best lambda, best alpha, and cross validation errors

Examples

```
CV_logisticc(X, y, lam = seq(0.1, 2, 0.1), alpha = c(1.1, 1.9, 0.1), penalty = 'bridge', method = 'MM', vec = c
```

data_gen *Normal Linear Data Generator*

Description

True beta values are generated from p independent draws from N(0, 1/p) distribution. X_-1 are n independent draws from (p - 1) multivariate normal N(0, Sigma) where Sigma has (j, k) entry theta^abs(j - k).

Y is then generated using the X = (1, X_-1) and true beta values with an iid error term that follows distribution N(0, var). We can specify the desired number of replications (reps).

Usage

```
data_gen(n, p, theta, var = 0.5, reps = 200)
```

Arguments

n	desired sample size
p	desired dimension
theta	parameter used to generate covariance matrix
var	variance of generated y values
reps	number of replications

Value

generated design matrix (X), response values (Y)(matrix if reps > 1), true beta values

Examples

```
data_gen(1000, 10, 0.5)
```

diagnostic *Diagnostic*

Description

This function simply streamlines the process of creating diagnostic plots with ggplot

Usage

```
diagnostic(data., x., y.)
```

Arguments

data.	data frame
x.	x-axis
y.	y-axis

Value

a residual plot and QQ plot

Examples

```
diagnostic(iris, Sepal.Length, Sepal.Width)
```

`dsearch` *Dichotomous search*

Description

Minimizes a univariate strictly quasiconvex function over the interval [a, b]. This is augmented code from Adam Rothman's STAT 8054 course (2017).

Usage

```
dsearch(g, a, b, L = 1e-07, eps = (L/2.1), quiet = FALSE)
```

Arguments

`g`	the function to minimize, where g(u, ...) is the function evaluated at u.
`a`	left endpoint of the initial interval of uncertainty.
`b`	right endpoint of the initial interval of uncertainty.
`L`	the maximum length of the final interval of uncertainty.
`eps`	search parameter, must be less than L/2
`quiet`	should the function stay quiet?
`...`	additional argument specifications for g

Value

returns the midpoint of the final interval of uncertainty.

Examples

```
dsearch(g, -10, 10, quiet = T)
```

gradient_IRLS_logistic

Gradient of Logistic Regression (IRLS)

Description

Computes the gradient of logistic regression (optional ridge regularization term). We use this to determine if the KKT conditions are satisfied. This function is to be used with the 'IRLS' function.

Usage

```
gradient_IRLS_logistic(betas, X, y, lam = 0, vec)
```

Arguments

betas	beta estimates (includes intercept)
X	matrix or data frame
y	response vector of 0,1
lam	tuning parameter for ridge regularization term
vec	vector to specify which coefficients will be penalized

Value

returns the gradient

Examples

```
gradient_IRLS_logistic(betas, X, y, lam = 0.1, penalty = 'ridge')
```

gradient_IRLS_logisticc

Gradient of Logistic Regression (IRLS) (c++)

Description

Computes the gradient of logistic regression (optional ridge regularization term). We use this to determine if the KKT conditions are satisfied. This function is to be used with the 'IRLSc' function.

Usage

```
gradient_IRLS_logisticc(betas, X, y, lam = 0, vec = 0L)
```

Arguments

betas	estimates (includes intercept)
X	matrix
y	response vector of 0,1
lam	tuning parameter for ridge regularization term
vec	vector to specify which coefficients will be penalized

Value

returns the gradient

Examples

```
gradient_IRLS_logistic(betas, X, y, lam = 0.1, vec = c(0,1,1,1))
```

`gradient_linearc` *Gradient of Linear Regression (c++)*

Description

Computes the gradient of linear regression (optional ridge regularization term). This function is to be used with the 'SVDc' function.

Usage

```
gradient_linearc(betas, X, y, lam = 0, weights = 0L, intercept = TRUE)
```

Arguments

`X`	matrix
`y`	response vector of 0,1
`lam`	tuning parameter for ridge regularization term
`weights`	option vector of weights for weighted least squares
`intercept`	add column of ones if not already present. Defaults to TRUE
`beta`	estimates (includes intercept)

Value

returns the gradient

Examples

```
gradient_linearc(betas, X, y, lam = 0.1, weights = rep(1,150), intercept = TRUE)
```

gradient_MM_linearc *Gradient of Linear Regression (MM) (c++)*

Description

Computes the gradient of linear regression (optional ridge and bridge regularization terms). We use this to determine if the KKT conditions are satisfied. This function is to be used with the 'MM_linearc' function.

Usage

```
gradient_MM_linearc(betas, X, y, lam = 0, alpha = 1.5, gamma = 1,
  weights = 0L, vec = 0L)
```

Arguments

betas	beta estimates (includes intercept)
X	matrix
y	response vector of 0,1
lam	tuning parameter for ridge regularization term
alpha	optional tuning parameter for bridge regularization term. Defaults to 'alpha = 1.5'
gamma	indicator function. 'gamma = 1' for ridge, 'gamma = 0' for bridge. Defaults to 'gamma = 1'
vec	vector to specify which coefficients will be penalized

Value

returns the gradient

Examples

```
gradient_MM_linearc(betas, X, y, lam = 0.1, alpha = 1.5, penalty = 'bridge')
```

gradient_MM_logistic *Gradient of Logistic Regression (MM)*

Description

Computes the gradient of logistic regression (optional ridge regularization term). We use this to determine if the KKT conditions are satisfied. This function is to be used with the 'MM' function.

Usage

```
gradient_MM_logistic(betas, X, y, lam = 0, alpha = 1.5, gamma = 1, vec)
```

Arguments

betas	beta estimates (includes intercept)
X	matrix or data frame
y	response vector of 0,1
lam	tuning parameter for ridge regularization term
alpha	optional tuning parameter for bridge regularization term. Defaults to 'alpha = 1.5'
gamma	indicator function. 'gamma = 1' for ridge, 'gamma = 0' for bridge. Defaults to 'gamma = 1'
vec	vector to specify which coefficients will be penalized

Value

returns the gradient

Examples

```
gradient_MM_logistic(betas, X, y, lam = 0.1, alpha = 1.5, penalty = 'bridge')
```

---

gradient_MM_logisticc *Gradient of Logistic Regression (MM) (c++)*

Description

Computes the gradient of logistic regression (optional ridge and bridge regularization terms). We use this to determine if the KKT conditions are satisfied. This function is to be used with the 'MMc' function.

Usage

```
gradient_MM_logisticc(betas, X, y, lam = 0, alpha = 1.5, gamma = 1,
  vec = 0L)
```

Arguments

betas	beta estimates (includes intercept)
X	matrix
y	response vector of 0,1
lam	tuning parameter for ridge regularization term. Defaults to 'lam = 0'
alpha	optional tuning parameter for bridge regularization term. Defaults to 'alpha = 1.5'
gamma	indicator function. 'gamma = 1' for ridge, 'gamma = 0' for bridge. Defaults to 'gamma = 1'
vec	vector to specify which coefficients will be penalized

Value

returns the gradient

Examples

```
gradient_MM_logistic(betas, X, y, lam = 0.1, alpha = 1.5, vec = c(0,1,1,1))
```

IRLS *Iterative Re-Weighted Least Squares*

Description

Computes the logistic regression coefficient estimates using the iterative re-weighted least squares (IRLS) algorithm. This function is to be used with the 'logisticr' function.

Usage

```
IRLS(X, y, lam = 0, intercept = TRUE, tol = 10^(-5), maxit = 1e+05, vec)
```

Arguments

X	matrix or data frame
y	matrix or vector of response 0,1
lam	tuning parameter for regularization term
intercept	Defaults to TRUE
tol	tolerance - used to determine algorithm convergence
maxit	maximum iterations
vec	optional vector to specify which coefficients will be penalized
betas	beta estimates (includes intercept)

Value

returns beta estimates (includes intercept), total iterations, and gradients.

Examples

```
IRLS(X, y, n.list = c(rep(1, n)), lam = 0.1, alpha = 1.5)
```

IRLSc *Iterative Re-Weighted Least Squares (c++)*

Description

Computes the logistic regression coefficient estimates using the iterative re-weighted least squares (IRLS) algorithm. This function is to be used with the 'logisticc' function.

Usage

```
IRLSc(X, y, lam = 0, penalty = "none", intercept = TRUE, tol = 1e-05,
  maxit = 1e+05, vec = 0L, init = 0L)
```

Arguments

X	matrix
y	matrix or vector of response 0,1
lam	tuning parameter for regularization term
penalty	choose from c('none', 'ridge'). Defaults to 'none'
intercept	Defaults to TRUE
tol	tolerance - used to determine algorithm convergence
maxit	maximum iterations
vec	optional vector to specify which coefficients will be penalized
betas	beta estimates (includes intercept)

Value

returns beta estimates (includes intercept), total iterations, and gradients.

Examples

```
IRLSc(X, y, lam = 0.1, penalty = 'ridge', vec = c(0,1,1,1))
```

---

kfold *K fold (c++)*

Description

creates vector of shuffled indices

Usage

```
kfold(n, K)
```

Arguments

n	number of eleemtns
K	number of folds

Value

returns vector

Examples

```
kfold(10, 3)
```

linearc *Linearc (c++)*

Description

Computes the linear regression coefficient estimates (ridge and bridge penalization and weights, optional)

Usage

```
linearc(X, y, lam = 0, alpha = 1.5, penalty = "none", weights = 0L,
  intercept = TRUE, kernel = FALSE, method = "SVD", tol = 1e-05,
  maxit = 1e+05, vec = 0L, init = 0L)
```

Arguments

X	matrix
y	matrix
lam	optional tuning parameter for ridge regularization term. Defaults to 'lam = 0'
alpha	optional tuning parameter for bridge regularization term. Defaults to 'alpha = 1.5'
penalty	choose from c('none', 'ridge', 'bridge'). Defaults to 'none'
weights	optional vector of weights for weighted least squares
intercept	add column of ones if not already present. Defaults to TRUE
kernel	use linear kernel to compute ridge regression coefficeients. Defaults to TRUE when p » n (for 'SVD')
method	optimization algorithm. Choose from 'SVD' or 'MM'. Defaults to 'SVD'
tol	tolerance - used to determine algorithm convergence for 'MM'. Defaults to 10^-5
maxit	maximum iterations for 'MM'. Defaults to 10^5
vec	optional vector to specify which coefficients will be penalized
init	optional initialization for MM algorithm

Value

returns the coefficient estimates

Examples

```
Weighted ridge regression
library(dplyr)
X = dplyr::select(iris, -c(Species, Sepal.Length))
y = dplyr::select(iris, Sepal.Length)
linearc(X, y, lam = 0.1, penalty = 'ridge', weights = rep(1:150), vec = c(0,1,1,1))

Kernelized ridge regression
linearc(X, y, lam = 0.1, penalty = 'ridge', kernel = T, vec = c(0,1,1,1))
```

linearr *Linear*

Description

Computes the linear regression coefficient estimates (ridge-penalization and weights, optional)

Usage

```
linearr(X, y, lam = seq(0, 2, 0.1), alpha = 1.5, penalty = "none",
  weights = NULL, intercept = TRUE, kernel = FALSE, method = "SVD",
  tol = 1e-05, maxit = 1e+05, vec = NULL, init = 1, K = 5)
```

Arguments

X	matrix or data frame
y	matrix or data frame of response values
lam	optional tuning parameter for ridge regularization term. If passing a list of values, the function will choose the optimal value based on K-fold cross validation. Defaults to 'lam = seq(0, 2, 0.1)'
alpha	optional tuning parameter for bridge regularization term. If passing a list of values, the function will choose the optimal value based on K-fold cross validation. Defaults to 'alpha = 1.5'
penalty	choose from c('none', 'ridge', 'bridge'). Defaults to 'none'
weights	optional vector of weights for weighted least squares
intercept	add column of ones if not already present. Defaults to TRUE
kernel	use linear kernel to compute ridge regression coefficeients. Defaults to TRUE when p » n (for 'SVD')
method	optimization algorithm. Choose from 'SVD' or 'MM'. Defaults to 'SVD'
tol	tolerance - used to determine algorithm convergence for 'MM'. Defaults to 10^-5
maxit	maximum iterations for 'MM'. Defaults to 10^5
vec	optional vector to specify which coefficients will be penalized
init	optional initialization for MM algorithm
K	specify number of folds for cross validation, if necessary

Value

returns the selected tuning parameters, coefficient estimates, MSE, and gradients

Examples

```
Weighted ridge regression
library(dplyr)
X = dplyr::select(iris, -c(Species, Sepal.Length))
y = dplyr::select(iris, Sepal.Length)
linearr(X, y, lam = 0.1, penalty = 'ridge', weights = rep(1:150))
```

```
Kernelized ridge regression
linearr(X, y, lam = 0.1, penalty = 'ridge', kernel = T)
```

---

logisticc *Logistic Regression (c++)*

Description

Computes the coefficient estimates for logistic regression. ridge regularization and bridge regularization optional.

Usage

```
logisticc(X, y, lam = 0, alpha = 1.5, penalty = "none",
  intercept = TRUE, method = "IRLS", tol = 1e-05, maxit = 1e+05,
  vec = 0L, init = 0L)
```

Arguments

X	matrix
y	matrix or vector of response values 0,1
lam	optional tuning parameter for ridge regularization term. Defaults to 'lam = 0'
alpha	optional tuning parameter for bridge regularization term. Defaults to 'alpha = 1.5'
penalty	choose from c('none', 'ridge', 'bridge'). Defaults to 'none'
intercept	Defaults to TRUE
method	optimization algorithm. Choose from 'IRLS' or 'MM'. Defaults to 'IRLS'
tol	tolerance - used to determine algorithm convergence. Defaults to 1e-5
maxit	maximum iterations. Defaults to 1e5
vec	optional vector to specify which coefficients will be penalized
init	optional initialization for MM algorithm

Value

returns beta estimates (includes intercept), total iterations, and gradients.

Examples

```
Logistic Regression
library(dplyr)
X = as.matrix(dplyr::select(iris, -Species))
y = as.matrix(dplyr::select(iris, Species))
y = ifelse(y == 'setosa', 1, 0)
logisticc(X, y, vec = c(0,1,1,1))

ridge Logistic Regression with IRLS
logisticc(X, y, lam = 0.1, penalty = 'ridge', vec = c(0,1,1,1))

ridge Logistic Regression with MM
```

```
logisticc(X, y, lam = 0.1, penalty = 'ridge', method = 'MM', vec = c(0,1,1,1))

bridge Logistic Regression
logisticc(X, y, lam = 0.1, alpha = 1.5, penalty = 'bridge', method = 'MM', vec = c(0,1,1,1))
```

`logisticr` *Logistic Regression*

Description

Computes the coefficient estimates for logistic regression. ridge regularization and bridge regularization optional.

Usage

```
logisticr(X, y, lam = seq(0, 2, 0.1), alpha = 1.5, penalty = "none",
  intercept = TRUE, method = "IRLS", tol = 1e-05, maxit = 1e+05,
  vec = NULL, init = 1, criteria = "logloss", K = 5)
```

Arguments

`X`	matrix or data frame
`y`	matrix or vector of response values 0,1
`lam`	optional tuning parameter(s) for ridge regularization term. If passing a list of values, the function will choose optimal value based on K-fold cross validation. Defaults to 'lam = seq(0, 2, 0.1)'
`alpha`	optional tuning parameter for bridge regularization term. If passing a list of values, the function will choose the optimal value based on K-fold cross validation. Defaults to 'alpha = 1.5'
`penalty`	choose from c('none', 'ridge', 'bridge'). Defaults to 'none'
`intercept`	Defaults to TRUE
`method`	optimization algorithm. Choose from 'IRLS' or 'MM'. Defaults to 'IRLS'
`tol`	tolerance - used to determine algorithm convergence. Defaults to 10^-5
`maxit`	maximum iterations. Defaults to 10^5
`vec`	optional vector to specify which coefficients will be penalized
`init`	optional initialization for MM algorithm
`criteria`	specify the criteria for cross validation. Choose from c('mse', 'logloss', 'misclass'). Defauls to 'logloss'
`K`	specify number of folds for cross validation, if necessary

Value

returns selected tuning parameters, beta estimates (includes intercept), MSE, log loss, misclassification rate, total iterations, and gradients.

Examples

```
Logistic Regression
library(dplyr)
X = dplyr::select(iris, -Species)
y = dplyr::select(iris, Species)
y$Species = ifelse(y$Species == 'setosa', 1, 0)
logisticr(X, y)

ridge Logistic Regression with IRLS
logistir(X, y, lam = 0.1, penalty = 'ridge')

ridge Logistic Regression with MM
logisticr(X, y, lam = 0.1, penalty = 'ridge', method = 'MM')

bridge Logistic Regression
(Defaults to MM -- IRLS will return error)
logisticr(X, y, lam = 0.1, alpha = 1.5, penalty = 'bridge')
```

---

logitc *Logitc (c++)*

Description

Computes the logit for u

Usage

```
logitc(u)
```

Arguments

u	some number

Value

returns the logit of u

Examples

```
logit(X*beta)
```

logitr *Logit*

Description

Computes the logit for u

Usage

```
logitr(u)
```

Arguments

u	some number

Value

returns the logit of u

Examples

```
logit(X %*% beta)
```

MM *Majorize-Minimization function*

Description

This function utilizes the MM algorithm. It will be used to compute the logistic regression coefficient estimates. This function is to be used with the 'logisticr' function.

Usage

```
MM(X, y, lam = 0, alpha = 1.5, gamma = 1, intercept = TRUE,
  tol = 10^(-5), maxit = 1e+05, vec = NULL)
```

Arguments

X	matrix or data frame
y	matrix or vector of response 0,1
lam	optional tuning parameter for ridge regularization term. Defaults to 'lam = 0'
alpha	optional tuning parameter for bridge regularization term. Defaults to 'alpha = 1.5'
gamma	gamma indicator function. 'gamma = 1' for ridge, 'gamma = 0' for bridge. Defaults to 'gamma = 1'
intercept	defaults to TRUE
tol	tolerance - used to determine algorithm convergence
maxit	maximum iterations
vec	optional vector to specify which coefficients will be penalized

Value

returns beta estimates (includes intercept), total iterations, and gradients.

Examples

```
MM(X, y)
```

MMc *Logistic Majorize-Minimization function (c++)*

Description

This function utilizes the MM algorithm. It will be used to compute the logistic regression coefficient estimates. This function is to be used with the 'logisticc' function.

Usage

```
MMc(X, y, lam = 0, alpha = 1.5, gamma = 1, intercept = TRUE,
  tol = 1e-05, maxit = 1e+05, vec = 0L, init = 0L)
```

Arguments

X	matrix
y	matrix or vector of response 0,1
lam	optional tuning parameter for ridge regularization term. Defaults to 'lam = 0'
alpha	optional tuning parameter for bridge regularization term. Defaults to 'alpha = 1.5'
gamma	gamma indicator function. 'gamma = 1' for ridge, 'gamma = 0' for bridge. Defaults to 'gamma = 1'
intercept	defaults to TRUE
tol	tolerance - used to determine algorithm convergence
maxit	maximum iterations
vec	optional vector to specify which coefficients will be penalized
init	optional initialization for MM algorithm

Value

returns beta estimates (includes intercept), total iterations, and gradients.

Examples

```
MMc(X, y)
```

MM_linearc *Linear Majorize-Minimization function (c++)*

Description

This function utilizes the MM algorithm. It will be used to compute the linear regression coefficient estimates with optional regularization penalties. This function is to be used with the 'linearc' function.

Usage

```
MM_linearc(X, y, lam = 0, alpha = 1.5, gamma = 1, weights = 0L,
  intercept = TRUE, tol = 1e-05, maxit = 1e+05, vec = 0L, init = 0L)
```

Arguments

X	matrix
y	matrix
lam	optional tuning parameter for ridge regularization term. Defaults to 'lam = 0'
alpha	optional tuning parameter for bridge regularization term. Defaults to 'alpha = 1.5'
gamma	gamma indicator function. 'gamma = 1' for ridge, 'gamma = 0' for bridge. Defaults to 'gamma = 1'
intercept	defaults to TRUE
tol	tolerance - used to determine algorithm convergence
maxit	maximum iterations
vec	optional vector to specify which coefficients will be penalized
init	optional initialization for MM algorithm

Value

returns beta estimates (includes intercept), total iterations, and gradients.

Examples

```
MM_linearc(X, y)
```

multiplot *Multiple Plot*

Description

Taken from: http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/

Usage

```
multiplot(..., plotlist = NULL, file, cols = 1, layout = NULL)
```

Arguments

...	object can be passed in
cols	number of columns in layout
layout	a matrix specify the layout. If present, 'cols' is ignored

Value

plots

Examples

```
multiplot(p1, p2, cols = 1)
```

predict_linearc *Predict Linear Regression*

Description

Generates prediction for linear regression

Usage

```
predict_linearc(betas, X, y = 0L)
```

Arguments

betas	'linearr' object or matrix of betas
X	matrix of (new) observations
y	matrix of response values

Value

predictions and loss metrics

Examples

```
fitted = linearr(X, y, penalty = 'ridge')
predict_linearr(fitted$coefficients, X)
```

predict_linearr *Predict Linear Regression*

Description

Generates prediction for linear regression. Note that one can either input a 'linearr' object or a matrix of beta coefficients.

Usage

```
predict_linearr(object, X, y = NULL)
```

Arguments

object	'linearr' object or matrix of betas
X	matrix or data frame of (new) observations
y	optional, matrix or vector of response values

Value

predictions and loss metrics

Examples

```
fitted = linearr(X, y, lam = 0.1)
predict_linearr(fitted, X)
```

predict_logisticc *Predict Logistic Regression (c++)*

Description

Generates prediction for logistic regression

Usage

```
predict_logisticc(betas, X, y = 0L)
```

Arguments

betas	matrix of coefficientts
X	matrix of (new) observations
y	matrix of response values 0,1

Value

predictions and loss metrics

Examples

```
fitted = logisticr(X, y, lam = 0.1, penalty = 'ridge', method = 'MM')
predict_logisticr(fitted$coefficients, X)
```

---

predict_logisticr *Predict Logistic Regression*

Description

Generates prediction for logistic regression. Note that one can either input a 'logisticr' object or a matrix of beta coefficients.

Usage

```
predict_logisticr(object, X, y = NULL)
```

Arguments

object	'logisticr' object or matrix of betas
X	matrix or data frame of (new) observations
y	optional, matrix or vector of response values 0,1

Value

predictions and loss metrics

Examples

```
fitted = logisticr(X, y, lam = 0.1, penalty = 'ridge', method = 'MM')
predict_logisticr(fitted, X)
```

---

scatter *Scatter*

Description

This function simply streamlines the process of creating a scatterplot with ggplot

Usage

```
scatter(data., x., y.)
```

Arguments

data.	data frame
x.	x-axis
y.	y-axis

Value

a scatterplot

Examples

```
scatter(iris, Sepal.Length, Sepal.Width)
```

SVDc *Linear Singular Value Decomposition (c++)*

Description

Computes the logistic regression coefficient estimates using SVD. This function is to be used with the 'linearc' function.

Usage

```
SVDc(X, y, lam = 0, weights = 0L, intercept = TRUE, kernel = FALSE)
```

Arguments

X	matrix
y	matrix
lam	optional tuning parameter for ridge regularization term. Defaults to 'lam = 0'
weights	optional vector of weights for weighted least squares
intercept	add column of ones if not already present. Defaults to TRUE
kernel	use linear kernel to compute ridge regression coefficeients. Defaults to TRUE when p » n (for 'SVD')

Value

returns beta estimates (includes intercept) and gradients.

Examples

```
SVDc(X, y, lam = 0.1 weights = rep(1, 150))
```

tidy *Tidy*

Description

tidys package R code and updates package documentation. Directly uses Yihui Xie's 'formatR' package.

Usage

```
tidy()
```

Examples

```
tidy()
```

timeit *Time-It*

Description

Simple function that prints the computation time of a function

Usage

```
timeit(f)
```

Arguments

f	the function to time

Value

returns the elapsed time

Examples

```
timeit(lm(dist ~ speed, cars))
```

Index