Package: coneproj (via r-universe)

October 10, 2024

Type Package

Title Primal or Dual Cone Projections with Routines for Constrained Regression

Version 1.19

Date 2024-06-11

Author Mary C. Meyer and Xiyue Liao

Maintainer Xiyue Liao <xliao@sdsu.edu>

Description Routines doing cone projection and quadratic programming, as well as doing estimation and inference for constrained parametric regression and shape-restricted regression problems. See Mary C. Meyer (2013)<doi:10.1080/03610918.2012.659820> for more details.

License GPL (>= 2)

Depends R(>= 4.4.0)

Imports Rcpp (>= 0.10.4)

LinkingTo RcppArmadillo, Rcpp

NeedsCompilation yes

Suggests stats, graphics, grDevices, utils

Repository CRAN

Date/Publication 2024-06-11 20:10:06 UTC

Contents

`check_irred` *Routine for Checking Irreducibility*

Description

This routine checks the irreducibility of a set of edges, which are supposed to form the columns of a matrix. If a column is a positive linear combination of other columns, then it can be removed without affecting the problem; if there is a positive linear combination of columns of the matrix that equals the zero vector, then there is an implicit equality constraint in the matrix. In the former case, this routine delete the redundant columns and return a set of irreducible edges, while in the latter case, this routine will give the number of equality constraints in the matrix, and will leave this issue to the user to fix.

Usage

```
check_irred(mat)
```

Arguments

`mat`	A matrix whose columns are edges.

Value

`edge`	The edges kept after being checked about irreducibility.
`reducible`	A vector of the indice of the edges that are redundant in the original set of edges.
`equal`	A vector showing the number of equality constraints in the original set of edges.

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (1999) An extension of the mixed primal-dual bases algorithm to the case of more constraints than dimensions. *Journal of Statistical Planning and Inference* ***81***, 13–31.

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

Liao, X. and M. C. Meyer (2014) coneproj: An R package for the primal or dual cone projections with routines for constrained regression. *Journal of Statistical Software* ***61(12)***, 1–22.

Examples

```
## Not run:
  data(TwoDamat)
  dim(TwoDamat)
  ans <- check_irred(t(TwoDamat))

## End(Not run)
```

---

conc — *Specify a Concave Shape-Restriction in a SHAPEREG Formula*

Description

A symbolic routine to define that the mean vector is concave in a predictor in a formula argument to coneproj.

Usage

```
conc(x, numknots = 0, knots = 0, space = "E")
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`knots`	The knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`space`	A character specifying the method to create knots. It will not be used for a shape-restricted predictor without smoothing. The default value is "E".

Details

"conc" returns the vector "x" and imposes on it two attributes: name and shape.

The shape attribute is 4 ("concave"), and according to the value of the vector itself and this attribute, the cone edges of the cone generated by the constraint matrix, which constrains the relationship between the mean vector and "x" to be concave, will be made. The cone edges are a set of basis employed in the hinge algorithm.

Note that "conc" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in coneproj.

See references cited in this section for more details.

Value

The vector x with the shape attribute, i.e., shape: 4 ("concave").

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

Examples

```
x <- seq(-1, 2, by = 0.1)
n <- length(x)
y <- - x^2 + rnorm(n, .3)

# regress y on x under the shape-restriction: "concave"
ans <- shapereg(y ~ conc(x))

# make a plot
plot(x, y)
lines(x, fitted(ans), col = 2)
legend("bottomleft", bty = "n", "shapereg: concave fit", col = 2, lty = 1)
```

---

coneA *Cone Projection – Polar Cone*

Description

This routine implements the hinge algorithm for cone projection to minimize $||y - \theta||^2$ over the cone C of the form $\{\theta : A\theta \geq 0\}$.

Usage

```
coneA(y, amat, w = NULL, face = NULL, msg = TRUE)
```

Arguments

`y` A vector of length n.

`amat` A constraint matrix. The rows of amat must be irreducible. The column number of amat must equal the length of y.

`w` An optional nonnegative vector of weights of length n. If w is not given, all weights are taken to equal 1. Otherwise, the minimization of $(y-\theta)'w(y-\theta)$ over C is returned. The default is w = NULL.

`face` A vector of the positions of edges, which define the initial face for the cone projection. For example, when there are m cone edges, then face is a subset of $1,\ldots,m$. The default is face = NULL.

`msg` A logical flag. If msg is TRUE, then a warning message will be printed when there is a non-convergence problem; otherwise no warning message will be printed. The default is msg = TRUE

Details

The routine coneA dynamically loads a C++ subroutine "coneACpp". The rows of $-A$ are the edges of the polar cone Ω^o. This routine first projects y onto Ω^o to get the residual of the projection onto the constraint cone C, and then uses the fact that y is equal to the sum of the projection of y onto C and the projection of y onto Ω^o to get the estimation of θ. See references cited in this section for more details about the relationship between polar cone and constraint cone.

Value

`df` The dimension of the face of the constraint cone on which the projection lands.

`thetahat` The projection of y on the constraint cone.

`steps` The number of iterations before the algorithm converges.

`xmat` The rows of the matrix are the edges of the face of the polar cone on which the residual of the projection onto the constraint cone lands.

`face` A vector of the positions of edges, which define the face on which the final projection lands on. For example, when there are m cone edges, then face is a subset of $1,\ldots,m$.

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (1999) An extension of the mixed primal-dual bases algorithm to the case of more constraints than dimensions. *Journal of Statistical Planning and Inference* ***81***, 13–31.

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

Liao, X. and M. C. Meyer (2014) coneproj: An R package for the primal or dual cone projections with routines for constrained regression. *Journal of Statistical Software* ***61(12)***, 1–22.

See Also

coneB, constreg, qprog

Examples

```
# generate y
    set.seed(123)
    n <- 50
    x <- seq(-2, 2, length = 50)
    y <- - x^2 + rnorm(n)

# create the constraint matrix to make the first half of y monotonically increasing
# and the second half of y monotonically decreasing
    amat <- matrix(0, n - 1, n)
    for(i in 1:(n/2 - 1)){
       amat[i, i] <- -1; amat[i, i + 1] <- 1
    }
    for(i in (n/2):(n - 1)){
       amat[i, i] <- 1; amat[i, i + 1] <- -1
    }

# call coneA
    ans1 <- coneA(y, amat)
    ans2 <- coneA(y, amat, w = (1:n)/n)

# make a plot to compare the unweighted fit and the weighted fit
    par(mar = c(4, 4, 1, 1))
    plot(y, cex = .7, ylab = "y")
    lines(fitted(ans1), col = 2, lty = 2)
    lines(fitted(ans2), col = 4, lty = 2)
   legend("topleft", bty = "n", c("unweighted fit", "weighted fit"), col = c(2, 4), lty = c(2, 2))
    title("ConeA Example Plot")
```

---

coneB *Cone Projection – Constraint Cone*

Description

This routine implements the hinge algorithm for cone projection to minimize $||y - \theta||^2$ over the cone C of the form $\{\theta : \theta = v + \sum b_i \delta_i, i = 1, \ldots, m, b_1, \ldots, b_m \geq 0\}$, v is in V.

Usage

```
coneB(y, delta, vmat = NULL, w = NULL, face = NULL, msg = TRUE)
```

Arguments

- `y` A vector of length n.
- `delta` A matrix whose columns are the constraint cone edges. The columns of delta must be irreducible. Its row number must equal the length of y. No column of delta is contained in the column space of vmat.
- `vmat` A matrix whose columns are the basis of the linear space contained in the constraint cone. Its row number must equal the length of y. The columns of vmat must be linearly independent. The default is vmat = NULL
- `w` An optional nonnegative vector of weights of length n. If w is not given, all weights are taken to equal 1. Otherwise, the minimization of $(y-\theta)'w(y-\theta)$ over C is returned. The default is w = NULL.
- `face` A vector of the positions of edges, which define the initial face for the cone projection. For example, when there are m cone edges, then face is a subset of $1, \dots, m$. The default is face = NULL.
- `msg` A logical flag. If msg is TRUE, then a warning message will be printed when there is a non-convergence problem; otherwise no warning message will be printed. The default is msg = TRUE

Details

The routine coneB dynamically loads a C++ subroutine "coneBCpp".

Value

- `df` The dimension of the face of the constraint cone on which the projection lands.
- `yhat` The projection of y on the constraint cone.
- `steps` The number of iterations before the algorithm converges.
- `coefs` The coefficients of the basis of the linear space and the constraint cone edges contained in the constraint cone.
- `face` A vector of the positions of edges, which define the face on which the final projection lands on. For example, when there are m cone edges, then face is a subset of $1, \dots, m$.

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (1999) An extension of the mixed primal-dual bases algorithm to the case of more constraints than dimensions. *Journal of Statistical Planning and Inference* ***81***, 13–31.

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

Liao, X. and M. C. Meyer (2014) coneproj: An R package for the primal or dual cone projections with routines for constrained regression. *Journal of Statistical Software* ***61(12)***, 1–22.

See Also

coneA, shapereg

Examples

```
# generate y
    set.seed(123)
    n <- 50
    x <- seq(-2, 2, length = 50)
    y <- - x^2 + rnorm(n)

# create the edges of the constraint cone to make the first half of y monotonically increasing
# and the second half of y monotonically decreasing
    amat <- matrix(0, n - 1, n)
    for(i in 1:(n/2 - 1)){
       amat[i, i] <- -1; amat[i, i + 1] <- 1
    }
    for(i in (n/2):(n - 1)){
       amat[i, i] <- 1; amat[i, i + 1] <- -1
    }

# note that in coneB, the transpose of the edges of the constraint cone is provided
    delta <- crossprod(amat, solve(tcrossprod(amat)))

# make the basis of V
    vmat <- matrix(rep(1, n), ncol = 1)

# call coneB
    ans3 <- coneB(y, delta, vmat)
    ans4 <- coneB(y, delta, vmat, w = (1:n)/n)

# make a plot to compare the unweighted fit and weighted fit
    par(mar = c(4, 4, 1, 1))
    plot(y, cex = .7, ylab = "y")
    lines(fitted(ans3), col = 2, lty = 2)
    lines(fitted(ans4), col = 4, lty = 2)
   legend("topleft", bty = "n", c("unweighted fit", "weighted fit"), col = c(2, 4), lty = c(2, 2))
    title("ConeB Example Plot")
```

---

constreg *Constrained Parametric Regression*

Description

The least-squares regression model $y = X\beta + \varepsilon$ is considered, where the object is to find β to minimize $||y - X\beta||^2$, subject to $A\beta \geq 0$.

Usage

```
constreg(y, xmat, amat, w = NULL, test = FALSE, nloop = 1e+4)
```

Arguments

- `y` — A vector of length n.
- `xmat` — A full column-rank design matrix. The column number of xmat must equal the length of β.
- `amat` — A constraint matrix. The rows of amat must be irreducible. The column number of amat must equal the length of β.
- `w` — An optional nonnegative vector of weights of length n. If w is not given, all weights are taken to equal 1. Otherwise, the minimization of $(y - X\beta)'w(y - X\beta)$ over C is returned. The default is w = NULL.
- `test` — A logical scalar. If test == TRUE, then the p-value for the test $H_0 : \beta$ is in V versus $H_1 : \beta$ is in C is returned. C is the constraint cone of the form $\{\beta : A\beta \geq 0\}$, and V is the null space of A. The default is test = FALSE.
- `nloop` — The number of simulations used to get the p-value for the E_{01} test. The default is 1e+4.

Details

The hypothesis test $H_0 : \beta$ is in V versus $H_1 : \beta$ is in C is an exact one-sided test, and the test statistic is $E_{01} = (SSE_0 - SSE_1)/SSE_0$, which has a mixture-of-betas distribution when H_0 is true and ε is a vector following a standard multivariate normal distribution with mean 0. The mixing parameters are found through simulations. The number of simulations used to obtain the mixing distribution parameters for the test is 10,000. Such simulations usually take some time. For the "FEV" data set used as an example in this section, whose sample size is 654, the time to get a p-value is roughly 6 seconds.

The constreg function calls coneA for the cone projection part.

Value

- `constr.fit` — The constrained fit of y given that β is in the cone C of the form $\{\beta : A\beta \geq 0\}$.
- `unconstr.fit` — The unconstrainted fit, i.e., the least-squares regression of y on the space spanned by X.
- `pval` — The p-value for the hypothesis test $H_0 : \beta$ is in V versus $H_1 : \beta$ is in C. The constraint cone C has the form $\{\beta : A\beta \geq 0\}$ and V is the null space of A. If test == TRUE, a p-value is returned. Otherwise, the test is skipped and no p-value is returned.
- `coefs` — The estimated constrained parameters, i.e., the estimation of the vector β.

Note

In the 3D plot of the "FEV" example, it is shown that the unconstrained fit increases as "age" increases when "height" is large, but decreases as "age" increases when "height" is small. This does not make sense, since "FEV" should not decrease with respect to "age" given any value of "height". The constrained fit avoids this situation by keeping the fit of "FEV" non-decreasing with respect to "age".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Brunk, H. D. (1958) On the estimation of parameters restricted by inequalities. *The Annals of Mathematical Statistics* ***29 (2)***, 437–454.

Raubertas, R. F., C.-I. C. Lee, and E. V. Nordheim (1986) Hypothesis tests for normals means constrained by linear inequalities. *Communications in Statistics - Theory and Methods* ***15 (9)***, 2809–2833.

Meyer, M. C. and J. C. Wang (2012) Improved power of one-sided tests. *Statistics and Probability Letters* ***82***, 1619–1622.

Liao, X. and M. C. Meyer (2014) coneproj: An R package for the primal or dual cone projections with routines for constrained regression. *Journal of Statistical Software* ***61(12)***, 1–22.

See Also

coneA

Examples

```
# load the FEV data set
    data(FEV)

# extract the variables
    y <- FEV$FEV
    age <- FEV$age
    height <- FEV$height
    sex <- FEV$sex
    smoke <- FEV$smoke

# scale age and height
    scale_age <- (age - min(age)) / (max(age) - min(age))
    scale_height <- (height - min(height)) / (max(height) - min(height))

# make xmat
    xmat <- cbind(1, scale_age, scale_height, scale_age * scale_height, sex, smoke)

# make the constraint matrix
    amat <- matrix(0, 4, 6)
    amat[1, 2] <- 1; amat[2, 2] <- 1; amat[2, 4] <- 1
    amat[3, 3] <- 1; amat[4, 3] <- 1; amat[4, 4] <- 1

# call constreg to get constrained coefficient estimates
    ans1 <- constreg(y, xmat, amat)
    bhat1 <- coef(ans1)

# call lm to get unconstrained coefficient estimates
    ans2 <- lm(y ~ xmat[,-1])
    bhat2 <- coef(ans2)
```

```
# create a 3D plot to show the constrained fit and the unconstrained fit
    n <- 25
    xgrid <- seq(0, 1, by = 1/n)
    ygrid <- seq(0, 1, by = 1/n)
    x1 <- rep(xgrid, each = length(ygrid))
    x2 <- rep(ygrid, length(xgrid))
    xinterp <- cbind(x1, x2)
    xmatp <- cbind(1, xinterp, x1 * x2, 0, 0)

    thint1 <- crossprod(t(xmatp), bhat1)
    A1 <- matrix(thint1, length(xgrid), length(ygrid), byrow = TRUE)
    thint2 <- crossprod(t(xmatp), bhat2)
    A2 <- matrix(thint2, length(xgrid), length(ygrid), byrow = TRUE)

    par(mfrow = c(1, 2))
    par(mar = c(4, 1, 1, 1))
    persp(xgrid, ygrid, A1, xlab = "age", ylab = "height",
    zlab = "FEV", theta = -30)
    title("Constrained Fit")

    par(mar = c(4, 1, 1, 1))
    persp(xgrid, ygrid, A2, xlab = "age", ylab = "height",
    zlab = "FEV", theta = -30)
    title("Unconstrained Fit")
```

---

conv *Specify a Convex Shape-Restriction in a SHAPEREG Formula*

Description

A symbolic routine to define that the mean vector is convex in a predictor in a formula argument to coneproj.

Usage

```
conv(x, numknots = 0, knots = 0, space = "E")
```

Arguments

x	A numeric predictor which has the same length as the response vector.
numknots	The number of knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
knots	The knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
space	A character specifying the method to create knots. It will not be used for a shape-restricted predictor without smoothing. The default value is "E".

Details

"conv" returns the vector "x" and imposes on it two attributes: name and shape.

The shape attribute is 3 ("convex"), and according to the value of the vector itself and this attribute, the cone edges of the cone generated by the constraint matrix, which constrains the relationship between the mean vector and "x" to be convex, will be made. The cone edges are a set of basis employed in the hinge algorithm.

Note that "conv" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in coneproj.

See references cited in this section for more details.

Value

The vector x with the shape attribute, i.e., shape: 3 ("convex").

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

Examples

```
# generate y
 x <- seq(-1, 2, by = 0.1)
 n <- length(x)
 y <- x^2 + rnorm(n, .3)

 # regress y on x under the shape-restriction: "convex"
 ans <- shapereg(y ~ conv(x))

 # make a plot
 plot(x, y)
 lines(x, fitted(ans), col = 2)
 legend("topleft", bty = "n", "shapereg: convex fit", col = 2, lty = 1)
```

---

cubic — *A Data Set for the Example of the Qprog Function*

Description

This data set is used for the example of the qprog function.

Usage

```
data(cubic)
```

Format

A data frame with 50 observations on the following 2 variables.

`x` The predictor vector.

`y` The response vector.

Details

We use the qprog function to fit a constrained cubic to this data set. The constraint is that the true regression is increasing, convex and nonnegative.

Source

STAT640 HW 14 given by Dr. Meyer.

`decr` *Specify a Decreasing Shape-Restriction in a SHAPEREG Formula*

Description

A symbolic routine to define that the mean vector is decreasing in a predictor in a formula argument to shapereg.

Usage

```
decr(x, numknots = 0, knots = 0, space = "E")
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`knots`	The knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`space`	A character specifying the method to create knots. It will not be used for a shape-restricted predictor without smoothing. The default value is "E".

Details

"decr" returns the vector "x" and imposes on it two attributes: name and shape.

The shape attribute is 2 ("decreasing"), and according to the value of the vector itself and this attribute, the cone edges of the cone generated by the constraint matrix, which constrains the relationship between the mean vector and "x" to be decreasing, will be made. The cone edges are a set of basis employed in the hinge algorithm.

Note that "decr" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in coneproj.

See references cited in this section for more details.

Value

The vector x with the shape attribute, i.e., shape: 2 ("decreasing").

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

See Also

decr.conc, decr.conv

Examples

```
data(cubic)

# extract x
x <- - cubic$x

# extract y
y <- cubic$y

# regress y on x with the shape restriction: "decreasing"
ans <- shapereg(y ~ decr(x))

# make a plot
par(mar = c(4, 4, 1, 1))
plot(x, y, cex = .7, xlab = "x", ylab = "y")
lines(x, fitted(ans), col = 2)
legend("topleft", bty = "n", "shapereg: decreasing fit", col = 2, lty = 1)
```

decr.conc — *Specify a Decreasing and Concave Shape-Restriction in a SHAPEREG Formula*

Description

A symbolic routine to define that the mean vector is decreasing and concave in a predictor in a formula argument to coneproj.

Usage

```
decr.conc(x, numknots = 0, knots = 0, space = "E")
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`knots`	The knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`space`	A character specifying the method to create knots. It will not be used for a shape-restricted predictor without smoothing. The default value is "E".

Details

"decr.conc" returns the vector "x" and imposes on it two attributes: name and shape.

The shape attribute is 8 ("decreasing and concave"), and according to the value of the vector itself and this attribute, the cone edges of the cone generated by the constraint matrix, which constrains the relationship between the mean vector and "x" to be decreasing and concave, will be made. The cone edges are a set of basis employed in the hinge algorithm.

Note that "decr.conc" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in coneproj.

See references cited in this section for more details.

Value

The vector x with the shape attribute, i.e., shape: 8 ("decreasing and concave").

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

See Also

incr.conv, incr

Examples

```
data(cubic)

# extract x
x <-  cubic$x

# extract y
y <- - cubic$y

# regress y on x with the shape restriction: "decreasing" and "concave"
ans <- shapereg(y ~ decr.conc(x))

# make a plot
par(mar = c(4, 4, 1, 1))
plot(x, y, cex = .7, xlab = "x", ylab = "y")
lines(x, fitted(ans), col = 2)
legend("bottomleft", bty = "n", "shapereg: decreasing and concave fit", col = 2, lty = 1)
```

---

decr.conv — *Specify a Decreasing and Convex Shape-Restriction in a SHAPEREG Formula*

Description

A symbolic routine to define that the mean vector is decreasing and convex in a predictor in a formula argument to coneproj.

Usage

```
decr.conv(x, numknots = 0, knots = 0, space = "E")
```

Arguments

x	A numeric predictor which has the same length as the response vector.
numknots	The number of knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
knots	The knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
space	A character specifying the method to create knots. It will not be used for a shape-restricted predictor without smoothing. The default value is "E".

Details

"decr.conv" returns the vector "x" and imposes on it two attributes: name and shape.

The shape attribute is 6 ("decreasing and convex"), and according to the value of the vector itself and this attribute, the cone edges of the cone generated by the constraint matrix, which constrains the relationship between the mean vector and "x" to be decreasing and convex, will be made. The cone edges are a set of basis employed in the hinge algorithm.

Note that "decr.conv" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in coneproj.

See references cited in this section for more details.

Value

The vector x with the shape attribute, i.e., shape: 6 ("decreasing and convex").

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

See Also

decr.conc, decr

Examples

```
data(cubic)

# extract x
x <- - cubic$x

# extract y
y <- cubic$y

# regress y on x with the shape restriction: "decreasing" and "convex"
ans <- shapereg(y ~ decr.conv(x))

# make a plot
par(mar = c(4, 4, 1, 1))
plot(x, y, cex = .7, xlab = "x", ylab = "y")
lines(x, fitted(ans), col = 2)
legend("bottomright", bty = "n", "shapereg: decreasing and convex fit", col = 2, lty = 1)
```

feet *Foot Measurements for Fourth Grade Children*

Description

This data set was collected by the first author in a fourth grade classroom in Ann Arbor, MI, October 1997. We use the shapereg function to make a shape-restricted fit to this data set. "Width" is a continuous response variable, "length" is a continuous predictor variable, and "sex" is a categorical covariate. The constraint is that "width" is increasing with respect to "length".

Usage

```
data(feet)
```

Format

A data frame with 39 observations on the following 8 variables.

`name` First name of child.

`month` Birth month.

`year` Birth year.

`length` Length of longer foot (cm).

`width` Width of longer foot (cm), measured at widest part of foot.

`sex` Boy or girl.

`foot` Foot measured (right or left).

`hand` Right- or left-handedness.

Source

Meyer, M. C. (2006) Wider Shoes for Wider Feet? *Journal of Statistics Education* ***Volume 14, Number 1***.

Examples

```
data(feet)
l <- feet$length
w <- feet$width
s <- feet$sex
plot(l, w, type = "n", xlab = "Foot Length (cm)", ylab = "Foot Width (cm)")
points(l[s == "G"], w[s == "G"], pch = 24, col = 2)
points(l[s == "B"], w[s == "B"], pch = 21, col = 4)
legend("topleft", bty = "n", c("Girl", "Boy"), pch = c(24, 21), col = c(2, 4))
title("Kidsfeet Width vs Length Scatterplot")
```

FEV *Forced Expiratory Volume*

Description

This data set consists of 654 observations on children aged 3 to 19. Forced Expiratory Volume (FEV), which is a measure of lung capacity, is the variable in interest. Age and height are two continuous predictors. Sex and smoke are two categorical predictors.

Usage

```
data(FEV)
```

Format

A data frame with 654 observations on the following 5 variables.

`age` Age of the 654 children.

`FEV` Forced expiratory volume(liters).

`height` Height(inches).

`sex` Female is 0. Male is 1.

`smoke` Nonsmoker is 0. Smoker is 1.

Source

Rosner, B. (1999) *Fundamentals of Biostatistics, 5th Ed., Pacific Grove, CA: Duxbur*.

Michael J. Kahn (2005) An Exhalent Problem for Teaching Statistics *Journal of Statistics Education* ***Volume 13, Number 2***.

---

incr *Specify an Increasing Shape-Restriction in a SHAPEREG Formula*

Description

A symbolic routine to define that the mean vector is increasing in a predictor in a formula argument to shapereg.

Usage

```
incr(x, numknots = 0, knots = 0, space = "E")
```

Arguments

- `x` — A numeric predictor which has the same length as the response vector.
- `numknots` — The number of knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
- `knots` — The knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
- `space` — A character specifying the method to create knots. It will not be used for a shape-restricted predictor without smoothing. The default value is "E".

Details

"incr" returns the vector "x" and imposes on it two attributes: name and shape.

The shape attribute is 1 ("increasing"), and according to the value of the vector itself and this attribute, the cone edges of the cone generated by the constraint matrix, which constrains the relationship between the mean vector and "x" to be increasing, will be made. The cone edges are a set of basis employed in the hinge algorithm.

Note that "incr" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in coneproj.

See references cited in this section for more details.

Value

The vector x with the shape attribute, i.e., shape: 1 ("increasing").

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

See Also

`incr.conc`, `incr.conv`

Examples

```
data(cubic)

# extract x
x <- cubic$x

# extract y
y <- cubic$y

# regress y on x with the shape restriction: "increasing"
```

```
ans <- shapereg(y ~ incr(x))

# make a plot
par(mar = c(4, 4, 1, 1))
plot(x, y, cex = .7, xlab = "x", ylab = "y")
lines(x, fitted(ans), col = 2)
legend("topleft", bty = "n", "shapereg: increasing fit", col = 2, lty = 1)
```

`incr.conc` *Specify an Increasing and Concave Shape-Restriction in a SHAPEREG Formula*

Description

A symbolic routine to define that the mean vector is increasing and concave in a predictor in a formula argument to coneproj.

Usage

```
incr.conc(x, numknots = 0, knots = 0, space = "E")
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`knots`	The knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`space`	A character specifying the method to create knots. It will not be used for a shape-restricted predictor without smoothing. The default value is "E".

Details

"incr.conc" returns the vector "x" and imposes on it two attributes: name and shape.

The shape attribute is 7 ("increasing and concave"), and according to the value of the vector itself and this attribute, the cone edges of the cone generated by the constraint matrix, which constrains the relationship between the mean vector and "x" to be increasing and concave, will be made. The cone edges are a set of basis employed in the hinge algorithm.

Note that "incr.conc" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in coneproj.

See references cited in this section for more details.

Value

The vector x with the shape attribute, i.e., shape: 7 ("increasing and concave").

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

See Also

incr.conv, incr

Examples

```
data(cubic)

# extract x
x <- - cubic$x

# extract y
y <- - cubic$y

# regress y on x with the shape restriction: "increasing" and "concave"
ans <- shapereg(y ~ incr.conc(x))

# make a plot
par(mar = c(4, 4, 1, 1))
plot(x, y, cex = .7, xlab = "x", ylab = "y")
lines(x, fitted(ans), col = 2)
legend("topleft", bty = "n", "shapereg: increasing and concave fit", col = 2, lty = 1)
```

---

incr.conv — *Specify an Increasing and Convex Shape-Restriction in a SHAPEREG Formula*

Description

A symbolic routine to define that the mean vector is increasing and convex in a predictor in a formula argument to coneproj.

Usage

```
incr.conv(x, numknots = 0, knots = 0, space = "E")
```

Arguments

- `x` A numeric predictor which has the same length as the response vector.
- `numknots` The number of knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
- `knots` The knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
- `space` A character specifying the method to create knots. It will not be used for a shape-restricted predictor without smoothing. The default value is "E".

Details

"incr.conv" returns the vector "x" and imposes on it two attributes: name and shape.

The shape attribute is 5 ("increasing and convex"), and according to the value of the vector itself and this attribute, the cone edges of the cone generated by the constraint matrix, which constrains the relationship between the mean vector and "x" to be increasing and convex, will be made. The cone edges are a set of basis employed in the hinge algorithm.

Note that "incr.conv" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in coneproj.

See references cited in this section for more details.

Value

The vector x with the shape attribute, i.e., shape: 5 ("increasing and convex").

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

See Also

incr.conc, incr

Examples

```
data(cubic)

# extract x
x <- cubic$x

# extract y
y <- cubic$y

# regress y on x with the shape restriction: "increasing" and "convex"
```

```
ans <- shapereg(y ~ incr.conv(x))

# make a plot
par(mar = c(4, 4, 1, 1))
plot(x, y, cex = .7, xlab = "x", ylab = "y")
lines(x, fitted(ans), col = 2)
legend("topleft", bty = "n", "shapereg: increasing and convex fit", col = 2, lty = 1)
```

---

qprog *Quadratic Programming*

Description

Given a positive definite n by n matrix Q and a constant vector c in R^n, the object is to find θ in R^n to minimize $\theta' Q\theta - 2c'\theta$ subject to $A\theta \geq b$, for an irreducible constraint matrix A. This routine transforms into a cone projection problem for the constrained solution.

Usage

```
qprog(q, c, amat, b, face = NULL, msg = TRUE)
```

Arguments

q	A n by n positive definite matrix.
c	A vector of length n.
amat	A m by n constraint matrix. The rows of amat must be irreducible.
b	A vector of length m. Its default value is 0.
face	A vector of the positions of edges, which define the initial face for the cone projection. For example, when there are m cone edges, then face is a subset of $1, \ldots, m$. The default is face = NULL.
msg	A logical flag. If msg is TRUE, then a warning message will be printed when there is a non-convergence problem; otherwise no warning message will be printed. The default is msg = TRUE

Details

To get the constrained solution to $\theta' Q\theta - 2c'\theta$ subject to $A\theta \geq b$, this routine makes the Cholesky decomposition of Q. Let $U'U = Q$, and define $\phi = U\theta$ and $z = U^{-1}c$, where U^{-1} is the inverse of U. Then we minimize $||z - \phi||^2$, subject to $B\phi \geq 0$, where $B = AU^{-1}$. It is now a cone projection problem with the constraint cone C of the form $\{\phi : B\phi \geq 0\}$. This routine gives the estimation of θ, which is U^{-1} times the estimation of ϕ.

The routine qprog dynamically loads a C++ subroutine "qprogCpp".

Value

df	The dimension of the face of the constraint cone on which the projection lands.
thetahat	A vector minimizing $\theta' Q\theta - 2c'\theta$.
steps	The number of iterations before the algorithm converges.
xmat	The rows of the matrix are the edges of the face of the polar cone on which the residual of the projection onto the constraint cone lands.
face	A vector of the positions of edges, which define the face on which the final projection lands on. For example, when there are m cone edges, then face is a subset of $1, \ldots, m$.

Author(s)

Mary C. Meyer and Xiyue Liao

References

Goldfarb, D. and A. Idnani (1983) A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming* ***27***, 1–33.

Fraser, D. A. S. and H. Massam (1989) A mixed primal-dual bases algorithm for regression under inequality constraints application to concave regression. *Scandinavian Journal of Statistics* ***16***, 65–74.

Fang,S.-C. and S. Puthenpura (1993) *Linear Optimization and Extensions*. Englewood Cliffs, New Jersey: Prentice Hall.

Silvapulle, M. J. and P. Sen (2005) *Constrained Statistical Inference*. John Wiley and Sons.

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

Liao, X. and M. C. Meyer (2014) coneproj: An R package for the primal or dual cone projections with routines for constrained regression. *Journal of Statistical Software* ***61(12)***, 1–22.

See Also

coneA

Examples

```
# load the cubic data set
    data(cubic)

# extract x
    x <- cubic$x

# extract y
    y <- cubic$y

# make the design matrix
    xmat <- cbind(1, x, x^2, x^3)
```

```
# make the q matrix
    q <- crossprod(xmat)

# make the c vector
    c <- crossprod(xmat, y)

# make the constraint matrix to constrain the regression to be increasing, nonnegative and convex
    amat <- matrix(0, 4, 4)
    amat[1, 1] <- 1; amat[2, 2] <- 1
    amat[3, 3] <- 1; amat[4, 3] <- 1
    amat[4, 4] <- 6
    b <- rep(0, 4)

# call qprog
    ans <- qprog(q, c, amat, b)

# get the constrained fit of y
    betahat <- fitted(ans)
    fitc <- crossprod(t(xmat), betahat)

# get the unconstrained fit of y
    fitu <- lm(y ~ x + I(x^2) + I(x^3))

# make a plot to compare fitc and fitu
    par(mar = c(4, 4, 1, 1))
    plot(x, y, cex = .7, xlab = "x", ylab = "y")
    lines(x, fitted(fitu))
    lines(x, fitc, col = 2, lty = 4)
   legend("topleft", bty = "n", c("constr.fit", "unconstr.fit"), lty = c(4, 1), col = c(2, 1))
    title("Qprog Example Plot")
```

shapereg *Shape-Restricted Regression*

Description

The regression model $y_i = f(t_i) + x_i'\beta + \varepsilon_i, i = 1, \ldots, n$ is considered, where the only assumptions about f concern its shape. The vector expression for the model is $y = \theta + X\beta + \varepsilon$. X represents a parametrically modelled covariate, which could be a categorical covariate or a linear term. The shapereg function allows eight shapes: increasing, decreasing, convex, concave, increasing-convex, increasing-concave, decreasing-convex, and decreasing-concave. This routine employs a single cone projection to find θ and β simultaneously.

Usage

```
shapereg(formula, data = NULL, weights = NULL, test = FALSE, nloop = 1e+4)
```

Arguments

`formula` A formula object which gives a symbolic description of the model to be fitted. It has the form "response ~ predictor". The response is a vector of length n. A predictor can be a non-parametrically modelled variable with a shape restriction or a parametrically modelled unconstrained covariate. In terms of a non-parametrically modelled predictor, the user is supposed to indicate the relationship between $E(y)$ and a predictor t in the following way:

- **incr(t):** $E(y)$ is increasing in t. See `incr` for more details.
- **decr(t):** $E(y)$ is decreasing in t. See `decr` for more details.
- **conc(t):** $E(y)$ is concave in t. See `conc` for more details.
- **conv(t):** $E(y)$ is convex in t. See `conv` for more details.
- **incr.conc(t):** $E(y)$ is increasing and concave in t. See `incr.conc` for more details.
- **decr.conc(t):** $E(y)$ is decreasing and concave in t. See `decr.conc` for more details.
- **incr.conv(t):** $E(y)$ is increasing and convex in t. See `incr.conv` for more details.
- **decr.conv(t):** $E(y)$ is decreasing and convex in t. See `decr.conv` for more details.

`data` An optional data frame, list or environment containing the variables in the model. The default is data = NULL.

`weights` An optional non-negative vector of "replicate weights" which has the same length as the response vector. If weights are not given, all weights are taken to equal 1. The default is weights = NULL.

`test` The test parameter given by the user.

`nloop` The number of simulations used to get the p-value for the E_{01} test. The default is 1e+4.

Details

This routine constrains θ in the equation $y = \theta + X\beta + \varepsilon$ by a shape parameter.

The constraint cone C has the form $\{\phi : \phi = v + \sum b_i\delta_i, i = 1, \ldots, m, b_1, \ldots, b_m \geq 0\}$, v is in V. The column vectors of X are in V, i.e., the linear space contained in the constraint cone.

The hypothesis test $H_0 : \phi$ is in V versus $H_1 : \phi$ is in C is an exact one-sided test, and the test statistic is $E_{01} = (SSE_0 - SSE_1)/(SSE_0)$, which has a mixture-of-betas distribution when H_0 is true and ε is a vector following a standard multivariate normal distribution with mean 0. The mixing parameters are found through simulations. The number of simulations used to obtain the mixing distribution parameters for the test is 10,000. Such simulations usually take some time. For the "feet" data set used as an example in this section, whose sample size is 39, the time to get a p-value is roughly between 4 seconds.

This routine calls coneB for the cone projection part.

Value

- `coefs` — The estimated coefficients for X, i.e., the estimation for the vector β. Note that even if the user does not provide a constant vector in X, the coefficient for the intercept will be returned.
- `constr.fit` — The shape-restricted fit over the constraint cone C of the form $\{\phi : \phi = v + \sum b_i\delta_i, i = 1, \ldots, m, b_1, \ldots, b_m \geq 0\}$, v is in V.
- `linear.fit` — The least-squares regression of y on V, i.e., the linear space contained in the constraint cone. If shape is 3 or shape is 4, V is spanned by X and t. Otherwise, it is spanned by X. X must be full column rank, and the matrix formed by combining X and t must also be full column rank.
- `se.beta` — The standard errors for the estimation of the vector β. The degree of freedom is returned by coneB and is multiplied by 1.5. Note that even if the user does not provide a constant vector in X, the standard error for the intercept will be returned.
- `pval` — The p-value for the hypothesis test $H_0 : \phi$ is in V versus $H_1 : \phi$ is in C. C is the constraint cone of the form $\{\phi : \phi = v+\sum b_i\delta_i, i = 1, \ldots, m, b_1, \ldots, b_m \geq 0\}$, v is in V, and V is the linear space contained in the constraint cone. If test == TRUE, a p-value is returned. Otherwise, the test is skipped and no p-value is returned.
- `pvals.beta` — The approximate p-values for the estimation of the vector β. A t-distribution is used as the approximate distribution. Note that even if the user does not provide a constant vector in X, the approximate p-value for the intercept will be returned.
- `test` — The test parameter given by the user.
- `SSE0` — The sum of squared residuals for the linear part.
- `SSE1` — The sum of squared residuals for the full model.
- `shape` — A number showing the shape constraint given by the user in a shapereg fit.
- `tms` — The terms objects extracted by the generic function *terms* from a shapereg fit.
- `zid` — A vector keeping track of the position of the parametrically modelled covariate.
- `vals` — A vector storing the levels of each variable used as a factor.
- `zid1` — A vector keeping track of the beginning position of the levels of each variable used as a factor.
- `zid2` — A vector keeping track of the end position of the levels of each variable used as a factor.
- `tnm` — The name of the shape-restricted predictor.
- `ynm` — The name of the response variable.
- `znms` — A vector storing the name of the parametrically modelled covariate.
- `is_param` — A logical scalar showing if or not a variable is a parametrically modelled covariate, which could be a factor or a linear term.
- `is_fac` — A logical scalar showing if or not a variable is a factor.
- `xmat` — A matrix whose columns represent the parametrically modelled covariate.
- `call` — The matched call.

Author(s)

Mary C. Meyer and Xiyue Liao

References

Raubertas, R. F., C.-I. C. Lee, and E. V. Nordheim (1986) Hypothesis tests for normals means constrained by linear inequalities. *Communications in Statistics - Theory and Methods* ***15 (9)***, 2809–2833.

Robertson, T., F. Wright, and R. Dykstra (1988) *Order Restricted Statistical Inference* New York: John Wiley and Sons.

Fraser, D. A. S. and H. Massam (1989) A mixed primal-dual bases algorithm for regression under inequality constraints application to concave regression. *Scandinavian Journal of Statistics* ***16***, 65–74.

Meyer, M. C. (2003) A test for linear vs convex regression function using shape-restricted regression. *Biometrika* ***90(1)***, 223–232.

Cheng, G.(2009) Semiparametric additive isotonic regression. *Journal of Statistical Planning and Inference* ***139***, 1980–1991.

Meyer, M.C.(2013a) Semiparametric additive constrained regression. *Journal of Nonparametric Statistics* ***25(3)***, 715–743.

Liao, X. and M. C. Meyer (2014) coneproj: An R package for the primal or dual cone projections with routines for constrained regression. *Journal of Statistical Software* ***61(12)***, 1–22.

See Also

coneB

Examples

```
# load the feet data set
    data(feet)

# extract the continuous and constrained predictor
    l <- feet$length

# extract the continuous response
    w <- feet$width

# extract the categorical covariate: sex
    s <- feet$sex

# make an increasing fit with test set as FALSE
    ans <- shapereg(w ~ incr(l) + factor(s))

# check the summary table
    summary(ans)

# make an increasing fit with test set as TRUE
    ans <- shapereg(w ~ incr(l) + factor(s), test = TRUE, nloop = 1e+3)
```

```
# check the summary table
    summary(ans)

# make a plot comparing the unconstrained fit and the constrained fit
    par(mar = c(4, 4, 1, 1))
    ord <- order(l)
    plot(sort(l), w[ord], type = "n", xlab = "foot length (cm)", ylab = "foot width (cm)")
    title("Shapereg Example Plot")

# sort l according to sex
    ord1 <- order(l[s == "G"])
    ord2 <- order(l[s == "B"])

# make the scatterplot of l vs w for boys and girls
    points(sort(l[s == "G"]), w[s == "G"][ord1], pch = 21, col = 1)
    points(sort(l[s == "B"]), w[s == "B"][ord2], pch = 24, col = 2)

# make an unconstrained fit to boys and girls
    fit <- lm(w ~ l + factor(s))

# plot the unconstrained fit
    lines(sort(l), (coef(fit)[1] + coef(fit)[2] * l + coef(fit)[3])[ord], lty = 2)
    lines(sort(l), (coef(fit)[1] + coef(fit)[2] * l)[ord], lty = 2, col = 2)
    legend(21.5, 9.8, c("boy","girl"), pch = c(24, 21), col = c(2, 1))

# plot the constrained fit
    lines(sort(l), (ans$constr.fit - ans$linear.fit + coef(ans)[1])[ord], col = 1)
    lines(sort(l), (ans$constr.fit - ans$linear.fit + coef(ans)[1] + coef(ans)[2])[ord], col = 2)
```

TwoDamat *A Two Dimensional Constraint Matrix*

Description

This is a two dimensional constraint matrix which will be used in the example for the check_irred routine.

Usage

```
data(TwoDamat)
```

Index