Package 'regtools'

August 25, 2019

Version 1.1.0

Title Regression and Classification Tools

Author Norm Matloff

Maintainer Norm Matloff <matloff@cs.ucdavis.edu>

Depends R (>= 3.5.0),FNN,mvtnorm,dummies,sandwich

Suggests knitr, rmarkdown

VignetteBuilder knitr

License GPL (>= 2)

Description Tools for linear, nonlinear and nonparametric regression and classification. Novel graphical methods for assessment of parametric models using nonparametric methods. One vs. All and All vs. All multiclass classification, optional class probabilities adjustment. Nonparametric regression (k-NN) for general dimension, local-linear option. Nonlinear regression with Eickert-White method for dealing with heteroscedasticity. Utilities for converting time series to rectangular form. Utilities for conversion between factors and indicator variables. Some code related to ``Statistical Regression and Classification: from Linear Models to Machine Learning", N. Matloff, 2017, CRC, ISBN 9781498710916.

URL https://github.com/matloff/regtools

BugReports https://github.com/matloff/regtools/issues

NeedsCompilation no

Repository CRAN

Date/Publication 2019-08-25 21:30:02 UTC

R topics documented:

`regtools-package` *Overview and Package Reference Guide*

Description

This package provides a broad collection of functions useful for regression and classification analysis.

Details

- k-NN: kmin, knnest, knntrn, preprocessx, meany, vary, loclin, predict, kmin, pwplot
- classification: classadjust, avalogtrn, avalogpred, ovalogtrn, ovalogpred, knntrn, predict, pwplot, nonparvsxplot
- regression diagnostics: parvsnonparplot, nonparvsxplot, nonparvarplot
- nonlinear regression: nlshc
- ridge regression: ridgelm, plot
- conversion between factors and dummies: dummiesToFactor, factorsToDummies, factorToDummies
- misc.: mm, tabletofakedf, unscale

`avalogtrn,avalogpred,ovalogtrn,ovalogpred,knntrn,predict.ovaknn,pwplot`
Classification with More Than 2 Classes

Description

One vs. All, All vs. All tools for multiclass classification, parametric and nonparametric.

Usage

```
ovalogtrn(m,trnxy,truepriors=NULL)
ovalogpred(coefmat,predx,probs=FALSE)
avalogtrn(m,trnxy)
avalogpred(m,coefmat,predx)
knntrn(y,xdata,m,k,truepriors=NULL)
## S3 method for class 'ovaknn'
predict(object,...)
classadjust(econdprobs,wrongratio,trueratio)
pwplot(y,x,k,pairs=combn(ncol(x),2),cexval=0.5,band=NULL)
```

Arguments

`x`	X data matrix or data frame.
`pairs`	Two-row matrix, column i of which is pair i of predictor variables to graph.
`cexval`	Symbol size for plotting.
`band`	If `band` is non-NULL, only points within `band`, say 0.1, of est. P(Y = 1) are displayed, for a contour-like effect.
`trnxy`	Data matrix (not a data frame), one data point per row, Y in the last column. Y must be numeric, 0,1,2,... `m-1`.
`object`	Needed for consistency with generic.
`...`	Needed for consistency with generic.
`y`	Vector of response variable data in the training set, with codes 0,1,...`m-1`.
`xdata`	X and associated neighbor indices. Output of `preprocessx`.
`coefmat`	Output from `ovalogtrn`.
`k`	Number of nearest neighbors.
`predx`	One data point to be predicted.
`m`	Number of classes in multiclass setting.
`econdprobs`	Estimated conditional class probabilities, given the predictors.
`wrongratio`	Incorrect, data-provenanced, p/(1-p), with p being the unconditional probability of a certain class.
`trueratio`	Same as `wrongratio`, but with the correct value.
`truepriors`	True unconditional class probabilities, typically obtained externally.
`probs`	If TRUE, return the estimated conditional probabilities of class membership.

Details

These functions do classification in the multiclass setting. During training, the `ova*` functions use the One vs. All method: For each class i, `as.integer(Y == i)` is regressed against the predictors, yielding a vector of beta coefficients for each class. For prediction, the new X is used with each such vector in the logit function, producing a conditional probability for Y == i over all classes i. The predicted value is then the i with maximal probability.

The `ava*` functions use the All vs. All method: During training, for each pair of classes i and j, i less than j, the data are restricted to cases having those classes. `as.integer(Y == i)` is regressed against the predictors, yielding a vector of beta coefficients for each pair. In prediction, the new X is used with each such vector in the logit function, producing a conditional probability for Y == i over all classes i and j, resulting in a "winner" between i and j. The predicted value is then the class with the most "wins."

In addition to logit, the k-Nearest Neighbor method is available. For this, `preprocessx` must first be called. In the logit case, All vs. All is also offered.

The functions `ovalogtrn`, `avalogtrn` and `knntrn` are used on the training set, and then fed into the prediction functions, `ovalogpred`, `avalogpred` and `predict.ovaknn`. The arguments for the latter are the output of `knntrn` and a matrix of prediction points (internally referred to as `predpts` in the code), one per row.

In `pwplot`, `y` must be a column of 0s and 1s. For each pair of predictor columns X12 in `x`, we compute estimated P(Y = 1) and P(Y = 1 | X12). If the latter is larger, plot a '1', else a '0'. The plot is intended to be helpful in exploring a relation between Y and X. For non-NULL `band`, a contour-like plot is made, showing where X12 changes from making Y = 1 less likely to more likely.

Value

The prediction functions, `ovalogpred`, `avalogpred` and `predict.ovaknn`, return the predicted classes for the points in `predx` or `predpts`.

The functions `ovalogtrn` and `avalogtrn` return the estimated logit coefficent vectors, one per column. There are `m` of them in the former case, `mm-1/2` in the latter, in which case the order of the R function `combin` is used.

The function `knntrn` returns a copy of the `xdata` input, but with an extra component added. The latter is the matrix of estimated regression function values; the element in row i, column j, is the probability that Y = j given that X = row i in the X data.

Author(s)

Norm Matloff

Examples

```
## Not run:
# toy example, kNN
set.seed(9999)
x <- runif(25)
y <- sample(0:2,25,replace=TRUE)
```

```
xd <- preprocessx(x,2,xval=TRUE)
kout <- knntrn(y,xd,m=3,k=2)
kout$regest  # row 2:  0.0,0.5,0.5
predict(kout,matrix(c(0.81,0.55,0.15),ncol=1))  # 0,1,2

# sim data, kNN
set.seed(9999)
n <- 1500
# within-grp cov matrix
cv <- rbind(c(1,0.2),c(0.2,1))
xy <- NULL
for (i in 1:3)
   xy <- rbind(xy,rmvnorm(n,mean=rep(i*2.0,2),sigma=cv))
y <- rep(0:2,each=n)
xy <- cbind(xy,y)
xdata <- preprocessx(xy[,-3],20)
oo <- knntrn(y,xdata,m=3,k=20)
predout <- predict(oo,xy[,-3])
mean(predout$predy == y)  # about 0.87

library(mlbench)
data(Vehicle)
xdata <- preprocessx(Vehicle[,-19],25)
kout <- knntrn(Vehicle$Class,xdata,k=25)
predict(kout,as.matrix(Vehicle[1,-19])) # predicted Y is 3

# UCI Letter Recognition data
data(LetterRecognition)
# prep data
lr <- LetterRecognition
# code Y values
lr[,1] <- as.numeric(lr[,1]) - 1
# training and test sets
lrtrn <- lr[1:14000,]
lrtest <- lr[14001:20000,]
# kNN
xdata <- preprocessx(lrtrn[,-1],50)
# without setting priors
trnout <- knntrn(lrtrn[,1],xdata,26,50)
ypred <- predict(trnout,as.matrix(lrtest[,-1]))
# how well did it work?
mean(ypred$predy == lrtest[,1])  # 0.86
# logit
ologout <- ovalogtrn(26,lr[,c(2:17,1)])
ypred <- ovalogpred(ologout,lr[,-1])
mean(ypred == lr[,1])  # only 0.73
# try quadratic terms
for (i in 2:17)
   lr <- cbind(lr,lr[,i]^2)
ologout1 <- ovalogtrn(26,lr[,c(2:33,1)])
ypred <- ovalogpred(ologout1,lr[,-1])
mean(ypred == lr[,1])  # increased to 0.81
```

```
library(mlbench)
data(PimaIndiansDiabetes)
pima <- PimaIndiansDiabetes
pima$diabetes <- as.integer(pima$diabetes == 'pos')
pwplot(pima$diabetes,pima[,1:8],25,pairs=cbind(c(2,3),c(2,6),c(6,8)),cex=0.8)
pwplot(pima$diabetes,pima[,1:8],25,pairs=cbind(c(2,3),c(2,6),c(6,8)),cex=0.8,band=0.05)


## End(Not run)
```

---

courseRecords *Records from several offerings of a certain course.*

Description

The data are in the form of an R list. Each element of the list corresponds to one offering of the course. Fields are: Class level; major (two different computer science majors, LCSI in Letters and Science and ECSE in engineering); quiz grade average (scale of 4.0, A+ counting as 4.3); homework grade average (same scale); and course letter grade.

---

currency *Pre-Euro Era Currency Fluctuations*

Description

From Wai Mun Fong and Sam Ouliaris, "Spectral Tests of the Martingale Hypothesis for Exchange Rates", Journal of Applied Econometrics, Vol. 10, No. 3, 1995, pp. 255-271. Weekly exchange rates against US dollar, over the period 7 August 1974 to 29 March 1989.

---

day *Bike sharing data.*

Description

This is the Bike Sharing dataset (day records only) from the UC Irvine Machine Learning Dataset Repository. Included here with permission of Dr. Hadi Fanaee.

english *English vocabulary data*

Description

The Stanford WordBank data on vocabulary acquisition in young children. The file consists of about 5500 rows. (There are many NA values, though, and only about 2800 complete cases.) Variables are age, birth order, sex, mother's education and vocabulary size.

factorsToDummies *Factor Conversion Utilities*

Description

Utilities from converting back and forth between factors and dummy variables.

Usage

```
factorToDummies(f,fname,omitLast=TRUE)
factorsToDummies(dfr,omitLast=TRUE)
dummiesToFactor(dms,inclLast=FALSE)
```

Arguments

f	A factor.
fname	A factor name.
dfr	A data frame.
omitLast	If TRUE, then generate only k-1 dummies from k factor levels.
dms	A data frame whose columns are dummy variables.
inclLast	If FALSE, then only k-1 dummies for k factor levels are provided.

Details

Many R users prefer to use R factors in their coding, or work with data that is of this type to begin with. On the other hand, many regression packages, e.g. **lars**, disallow factors. These utilities facilitate conversion from one form to another.

Value

The function factorToDummies returns a matrix of dummy variables, while factorsToDummies returns a new version of the input data frame, in which each factor is replaced by columns of dummies.

Author(s)

Norm Matloff

Examples

```
f <- as.factor(c('abc','de','f','de','abc'))
factorToDummies(f,'f')
# outputs
#      f.abc f.de
# [1,]     1    0
# [2,]     0    1
# [3,]     0    0
# [4,]     0    1
# [5,]     1    0
d <- data.frame(a=1:5,b=f)
factorsToDummies(d,omitLast=FALSE)
# outputs
#   a b.abc b.de
# 1 1     1    0
# 2 2     0    1
# 3 3     0    0
# 4 4     0    1
# 5 5     1    0
```

---

knnest,meany,vary,loclin,predict.knn,preprocessx,kmin,parvsnonparplot,nonparvsxplot,l1,l2

Nonparametric Regression and Classification

Description

Full set of tools for k-NN regression and classification, including both for direct usage and as tools for assessing the fit of parametric models.

Usage

```
knnest(y,xdata,k,nearf=meany)
preprocessx(x,kmax,xval=FALSE)
meany(predpt,nearxy)
vary(predpt,nearxy)
loclin(predpt,nearxy)
## S3 method for class 'knn'
predict(object,...)
kmin(y,xdata,lossftn=l2,nk=5,nearf=meany)

parvsnonparplot(lmout,knnout,cex=1.0)
nonparvsxplot(knnout,lmout=NULL)
nonparvarplot(knnout,returnPts=FALSE)
l2(y,muhat)
l1(y,muhat)
```

Arguments

y	Response variable data in the training set. Vector or matrix, the latter case for vector-valued response, e.g. multiclass classification.
x	X data, predictors, one row per data point, in the training set.
...	Needed for consistency with generic. See Details below for 'arguments.
xdata	X and associated neighbor indices. Output of preprocessx.
k	Number of nearest neighbors
object	Output of knnest.
predpt	One point on which to predict, as a vector.
nearxy	A set of X neighbors of a point.
nearf	Function to apply to the nearest neighbors of a point.
kmax	Maximal number of nearest neighbors to find.
xval	Cross-validation flag. If TRUE, then the set of nearest neighbors of a point will not include the point itself.
lossftn	Loss function to be used in cross-validation determination of "best" k.
nk	Number of values of k to try in cross-validation.
lmout	Output of lm.
knnout	Output of knnest.
cex	R parameter to control dot size in plot.
muhat	Vector of estimated regression function values.
returnPts	If TRUE, return matrix of plotted points.

Details

The knnest function does k-nearest neighbor regression function estimation, in any dimension, i.e. any number of predictor variables, and any number of response variables. This of course includes classification problems case; a scalar Y = 0,1 would represent two classes, with the regression function reducing to the conditional probability of class 1, given the predictors.

The preprocessx function does the prep work. For each row in x, the code finds the kmax closest rows to that row. By separating this computation from knnest, one can save a lot of overall computing time. If for instance one wants to try the number of nearest neighbors k at 25, 50 and 100, one can call preprocessx with kmax equal to 100, then reuse the results; in calling knnest for several values of k, we do not need to call preprocessx again. Setting xval to TRUE turns out cross-validation: the neighborhood of a point will not include the point itself; note that this is set in preprocessx, not in knnest.

One can specify various types of smoothing by proper specification of the nearf function. The default is meany, specifying the standard averaging of the neighbor Y values. Another possible choice is vary, specifying calculation of the sample variance of those Y values; this is useful in assessing heteroscedasticity in a linear model.

Another choice is to specify local linear smoothing by setting nearf to loclin. Here the value of the regression function at a point is predicted from a linear fit to the point's neighbors. This may be

especially helpful to counteract bias near the edges of the data. As in any regression fit, the number of predictors should be considerably less than the number of neighbors.

The X, i.e. predictor, data will be scaled by the code, so as to put all predictor variables on an equal footing. The scaling parameters will be recorded, and then applied later in prediction.

The function `predict.knn` uses the output of `knnest` to do regression estimation or prediction on new points. Since the output of `knnest` is of class `'knn'`, one invokes this function with the simpler `predict`. The second argument is the set of new points, named `predpts` within the code. It is specified as a matrix if there is more than one prediction point and more than one predictor variable; otherwise, use a vector.

A "1-NN" method is used here: Given a new point u whose Y value we wish to predict, the code finds the single closest row in the training set, and returns the previously-estimated regression function value at that row. If u needs to be scaled, specify `TRUE` in the third argument of `predict`; otherwise specify FALSE.

It can be shown that nearest-neighbor (or kernel) regression estimates are subject to substantial bias near the fringes of the data; the further away from the center of the data, the worse the bias. This can be mitigated by user specification that a local linear regression be applied, as follows: For each new point u to predict, the r closest X rows in the training set to u will be found, and a linear regression of the corresponding Y values against those X values will be computed. The result of that operation will be used to predict the Y value at the point u. The value of r is specified as the third argument in the call to `predict`; if left unspecified, the 1-NN method is used as described above, and it may be more accurate than the local-linear approach within the bulk of the data set.

The functions `ovaknntrn` and `ovaknnpred` are multiclass wrappers for `knnest` and `knnpred`. Here `y` is coded 0,1,...,`m`-1 for the `m` classes.

The tools here can be useful for fit assessment of parametric models. The `parvsnonparplot` function plots fitted values of parameteric model vs. kNN fitted, `nonparvsxplot` k-NN fitted values against each predictor, one by one.

The functions `l2` and `l1` are used to define L2 and L1 loss.

Value

The return value of `preprocessx` is an R list. Its `x` component is the scaled `x` matrix, with the scaling factors being recorded in the `scaling` component. The `idxs` component contains the indices of the nearest neighbors of each point in the predictor data, stored in a matrix with `nrow(x)` rows and `k` columns. Row i contains the indices of the nearest rows in `x` to row i of `x`. The first of these indices is for the closest point, then for the second-closest, and so on. If cross-validation is requrested (`xval = TRUE`, then any point will not be considered a neighbor of itself.

The `knnest` function returns an expanded version of `xdata`, with the expansion consisting of a new component `regest`, the estimated regression function values at the training set points.

The function `predict.knn` returns the predicted Y values at `predpts`. It is called simply via `predict`.

One can explore the effects of various numbers of nearest neighbors `k` through the `kmin` function. (This function should be cosidered experimental.) It will run `knnest` for the values of `k` specified in `nk`. If the latter is a number, the range 0 to `xdata$kmax` will be divided into `nk` equally subintervals, and the values of `k` used will then be the right endpoints of the subintervals. The function returns an R list, with the component `meanerrs` containing the cross-validated mean loss function values and `ks` containing the corresponding values of `k`; `plot.knn` then plots the former against the latter.

Author(s)

Norm Matloff

Examples

```
set.seed(9999)
x <- matrix(sample(1:100,30),ncol=3)
xd <- preprocessx(x[,1],2,TRUE)  # just 1 predictor
ko <- knnest(x[,2],xd,2)  # Y is x[,2]
ko$regest # 1st element = 74.5
predict(ko,matrix(76),TRUE)  # 47.5
ko <- knnest(x[,-1],xd,2)  # Y bivar
ko$regest # 1st row = (74.5,31.5)
predict(ko,matrix(76),TRUE)  # 47.5, 65.0

set.seed(9999)
xe <- matrix(rnorm(30000),ncol=3)
xe[,-3] <- xe[,-3] + 2
# xe is 2 predictors and epsilon
y <- xe %*% c(1,0.5,0.2)  # Y
x <- xe[,-3]  # X
xdata <- preprocessx(x,500)  # k as high as 500
zout <- knnest(y,xdata,200)
predict(zout,matrix(c(1,1),nrow=1),TRUE)  # about 1.55
predict(zout,rbind(c(1,1),c(2,1.2)),TRUE)  # about 1.55, 2.58
predict(zout,rbind(c(0,0)),TRUE)  # about 0.63

## Not run:
data(prgeng)
pe <- prgeng
# dummies for MS, PhD
pe$ms <- as.integer(pe$educ == 14)
pe$phd <- as.integer(pe$educ == 16)
# computer occupations only
pecs <- pe[pe$occ >= 100 & pe$occ <= 109,]
# for simplicity, let's choose a few predictors
pecs1 <- pecs[,c(1,7,9,12,13,8)]
# will predict wage income from age, gender etc.
# prepare nearest-neighbor data, k up to 50
xdata <- preprocessx(pecs1[,1:5],50)
zout <- knnest(pecs1[,6],xdata,50)  # k = 50
# find the est. mean income for 42-year-old women, 52 weeks worked, with
# a Master's
predict(zout,matrix(c(42,2,52,0,0),nrow=1),TRUE)  # 62106
# try k = 25; don't need to call preprocessx() again
zout <- knnest(pecs1[,6],xdata,25)
predict(zout,matrix(c(42,2,52,0,0),nrow=1),TRUE)  # 69104
# quite a difference; what k values are good?
kmout <- kmin(pecs1[,6],xdata) # at least 50
# what about a man?
zout <- knnest(pecs1[,6],xdata,50)
predict(zout,matrix(c(42,1,52,0,0),nrow=1),TRUE)  # 78588
```

```
# form training and test sets, fit on the former and predict on the
# latter
fullidxs <- 1:nrow(pecs1)
train <- sample(fullidxs,10000)
xdata <- preprocessx(pecs1[train,1:5],50)
trainout <- knnest(pecs1[train,6],xdata,50)
testout <- predict(trainout,as.matrix(pecs1[-train,-6]),TRUE)
# find mean abs. prediction error (about $25K)
mean(abs(pecs1[-train,6] - testout))
# examples of fit assessment
# look for nonlinear relations between Y and each X
nonparvsxplot(zout)  # keep hitting Enter for next plot
# there seem to be quadratic relations with age and wkswrkd, so add quad
# terms and run lm()
pecs2 <- pecs1
pecs2$age2 <- pecs1$age^2
pecs2$wks2 <- pecs1$wkswrkd^2
lmout2 <- lm(wageinc ~ .,data=pecs2)
# check parametric fit by comparing to kNN
parvsnonparplot(lmout2,zout)
# linear model line somewhat faint, due to large n;
# parametric model seems to overpredict at high end;
# to deal with faintness, reduce size of points
parvsnonparplot(lmout2,zout,cex=0.1)
# assess homogeneity of conditional variance
nonparvarplot(zout)
# hockey stick!

## End(Not run)

# Y vector-valued (3 classes)
# 3 clusters, equal wts, coded 0,1,2
n <- 1500
# within-grp cov matrix
cv <- rbind(c(1,0.2),c(0.2,1))
xy <- NULL
for (i in 1:3)
   xy <- rbind(xy,rmvnorm(n,mean=rep(i*2.0,2),sigma=cv))
y <- rep(0:2,each=n)
xy <- cbind(xy,dummy(y))
xdata <- preprocessx(xy[,-(3:5)],20) # X is xy[,1:2], k <= 20
ko <- knnest(xy[,3:5],xdata,20)
# find predicted Y for each data pt
mx <- apply(as.matrix(ko$regest),1,which.max) - 1
# overall correct classification rate
mean(mx == y)  # should be about 0.87
```

---

lmac,makeNA,coef.lmac,vcov.lmac,pcac,loglinac,tbltofakedf

Available Cases Method for Missing Data

Description

Various estimators that handle missing data via the Available Cases Method

Usage

```
lmac(xy,nboot=0)
makeNA(m,probna)
## S3 method for class 'lmac'
coef(object,...)
## S3 method for class 'lmac'
vcov(object,...)
pcac(indata,scale=FALSE)
loglinac(x,margin)
tbltofakedf(tbl)
```

Arguments

xy	Matrix or data frame, X values in the first columns, Y in the last column.
indata	Matrix or data frame.
x	Matrix or data frame, one column per variable.
nboot	If positive, number of bootstrap samples to take.
probna	Probability that an element will be NA.
scale	If TRUE, call cor instead of cov.
	Matrix or data frame, one column per variable.
tbl	An R table.
m	Number of synthetic NAs to insert.
object	Output from lmac.
...	Needed for consistency with generic function. Not used.
margin	A list of vectors specifying the model, as in loglin.

Details

The Available Cases (AC) approach applies to statistical methods that depend only on products of k of the variables, so that cases having non-NA values for those k variables can be used, as opposed to using only cases that are fully intact in all variables, the Complete Cases (CC) approach. In the case of linear regression, for instance, the estimated coefficients depend only on covariances between the variables (both predictors and response). This approach assumes thst the cases with missing values have the same distribution as the intact cases.

The lmac function forms OLS estimates as with lm, but applying AC, in contrast to lm, which uses the CC method.

The pcac function is an AC substitute for prcomp. The data is centered, corresponding to a fixed value of center = TRUE in prcomp. It is also scaled if scale is TRUE, corresponding scale. = TRUE in prcomp;. Due to AC, there is a small chance of negative eigenvalues, in which case stop will be called.

The loglinac function is an AC substitute for loglin. The latter takes tables as input, but loglinac takes the raw data. If you have just the table, use tbltofakedf to regenerate a usable data frame.

The makeNA function is used to insert random NA values into data, for testing purposes.

Value

For lmac, an object of class 'lmac', with components

- coefficients, as with lm; accessible directly or by calling coef, as with lm
- fitted.values, as with lm
- residuals, as with lm
- r2, (unadjusted) R-squared
- cov, for nboot >0 the estimated covariance matrix of the vector of estimated regression coefficients; accessible directly or by calling vcov, as with lm

For pcac, an R list, with components

- sdev, as with prcomp
- rotation, as with prcomp

For loglinac, an R list, with components

- param, estimated coefficients, as in loglin
- fit, estimated expected call counts, as in loglin

Author(s)

Norm Matloff

Examples

```
n <- 25000
w <- matrix(rnorm(2*n),ncol=2)  # x and epsilon
x <- w[,1]
y <- x + w[,2]
# insert some missing values
nmiss <- round(0.1*n)
x[sample(1:n,nmiss)] <- NA
nmiss <- round(0.2*n)
y[sample(1:n,nmiss)] <- NA
acout <- lmac(cbind(x,y))
coef(acout)  # should be near pop. values 0 and 1
```

ltrfreqs *Letter Frequencies*

Description

This is data consists of capital letter frequencies obtained at http://www.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.h tml

mlb *Major Leage Baseball player data set.*

Description

Heights, weights, ages etc. of major league baseball players. A new variable has been added, consolidating positions into Infielders, Outfielders, Catchers and Pitchers.

Included here with the permission of the UCLA Statistics Department.

mlens *MovieLens User Summary Data*

Description

The MovieLens dataset, http://grouplens.org/, is a standard example in the recommender systems literature. Here we give demographic data for each user, plus the mean rating and number of ratings. One may explore, for instance, the relation between ratings and age.

mm *Method of Moments, Including Possible Regression Terms*

Description

Method of Moments computation for almost any statistical problem that has derivatives with respect to theta. Capable of handling models that include parametric regression terms, but not need be a regression problem. (This is not *Generalized* Method of Moments; see the package **gmm** for the latter.)

Usage

```
mm(m,g,x,init=rep(0.5,length(m)),eps=0.0001,maxiters=1000)
```

Arguments

m	Vector of sample moments, "left-hand sides" of moment equations.
g	Function of parameter estimates, forming the "right-hand sides." This is a multivariate-valued function, of dimensionality equal to that of m.
init	Vector of initial guesses for parameter estimates. If components are named, these will be used as labels in the output.
eps	Convergence criterion.
maxiters	Maximum number of iterations.
x	Input data.

Details

Standard Newton-Raphson methods are used to solve for the parameter estimates, with numericDeriv being used to find the approximate derivatives.

Value

R list consisting of components tht, the vector of parameter estimates, and numiters, the number of iterations performed.

Author(s)

Norm Matloff

Examples

```
x <- rgamma(1000,2)
m <- c(mean(x),var(x))
g <- function(x,theta) {  # from theoretical properties of gamma distr.
   g1 <-  theta[1] / theta[2]
   g2 <-  theta[1] / theta[2]^2
   c(g1,g2)
}
# should output about 2 and 1
mm(m,g,x)

## Not run:
library(mfp)
data(bodyfat)
# model as a beta distribution
g <- function(x,theta) {
   t1 <- theta[1]
   t2 <- theta[2]
   t12 <- t1 + t2
   meanb <- t1 / t12
   m1 <- meanb
   m2 <- t1*t2 / (t12^2 * (t12+1))
   c(m1,m2)
}
```

```
x <- bodyfat$brozek/100
m <- c(mean(x),var(x))
# about 4.65 and 19.89
mm(m,g,x)

## End(Not run)
```

`newadult` *UCI adult income data set, adapted*

Description

This data set is adapted from the Adult data from the UCI Machine Learning Repository, which was in turn adapted from Census data on adult incomes and other demographic variables. The UCI data is used here with permission from Ronny Kohavi.

The variables are:

- `gt50`, which converts the original `>50K` variable to an indicator variable; 1 for income greater than $50,000, else 0
- `edu`, which converts a set of education levels to approximate number of years of schooling
- `age`
- `gender`, 1 for male, 0 for female
- `mar`, 1 for married, 0 for single

`nlshc` *Heteroscedastic Nonlinear Regression*

Description

Extension of `nls` to the heteroscedastic case.

Usage

```
nlshc(nlsout,type='HC')
```

Arguments

`nlsout`	Object of type 'nls'.
`type`	Eickert-White algorithm to use. See documentation for **nls**.

Details

Calls `nls` but then forms a different estimated covariance matrix for the estimated regression coefficients, applying the Eickert-White technique to handle heteroscedasticity. This then gives valid statistical inference in that setting.

Some users may prefer to use `nlsLM` of the package **minpack.lm** instead of `nls`. This is fine, as both functions return objects of class 'nls'.

Value

Estimated covariance matrix

Author(s)

Norm Matloff

References

Zeileis A (2006), Object-Oriented Computation of Sandwich Estimators. *Journal of Statistical Software*, **16**(9), 1–16, http://www.jstatsoft.org/v16/i09/.

Examples

```
# simulate data from a setting in which mean Y is
# 1 / (b1 * X1 + b2 * X2)
n <- 250
b <- 1:2
x <- matrix(rexp(2*n),ncol=2)
meany <- 1 / (x %*% b)  # reg ftn
y <- meany + (runif(n) - 0.5) * meany  # heterosced epsilon
xy <- cbind(x,y)
xy <- data.frame(xy)
# see nls() docs
nlout <- nls(X3 ~ 1 / (b1*X1+b2*X2),
   data=xy,start=list(b1 = 1,b2=1))
nlshc(nlout)
```

---

oliveoils *Italian olive oils data set.*

Description

Italian olive oils data set, as used in *Graphics of Large Datasets: Visualizing a Million*, by Antony Unwin, Martin Theus and Heike Hofmann, Springer, 2006. Included here with permission of Dr. Martin Theus.

prgeng *Silicon Valley programmers and engineers data*

Description

This data set is adapted from the 2000 Census (5% sample, person records). It is mainly restricted to programmers and engineers in the Silicon Valley area. (Apparently due to errors, there are some from other ZIP codes.)

There are three versions:

- `prgeng`, the original data, with categorical variables, e.g. Occupation, in their original codes
- `peDumms`, same but with categorical variables converted to dummies; due to the large number of levels the birth and PUMA data is not included
- `peFactors`, same but with categorical variables converted to factors

The variable codes, e.g. occupational codes, are available from the Census Bureau, at http://www.census.gov/prod/cen2000/doc/pums.pdf. (Short code lists are given in the record layout, but longer ones are in the appendix Code Lists.)

The variables are:

- `age`, with a U(0,1) variate added for jitter
- `cit`, citizenship; 1-4 code various categories of citizens; 5 means noncitizen (including permanent residents)
- `educ`: 01-09 code no college; 10-12 means some college; 13 is a bachelor's degree, 14 a master's, 15 a professional degree and 16 is a doctorate
- `occ`, occupation
- `birth`, place of birth
- `wageinc`, wage income
- `wkswrkd`, number of weeks worked
- `yrentry`, year of entry to the U.S. (0 for natives)
- `powpuma`, location of work
- `gender`, 1 for male, 2 for female

Usage

```
data(prgeng)
data(peDumms)
data(peFactors)
```

quizDocs *Course quiz documents*

Description

This data is suitable for NLP analysis.

This is an R list, 143 elements, one for each of 143 quizzes from my various courses. Each list element is a character vector, one vector element per line of the quiz.

The original documents were LaTeX files. They have been run through the `detex` utility to remove most LaTeX commands, as well as removing the LaTeX preambles separately.

The names of the list elements are the course names, as follows:

ECS 50: a course in machine organization

ECS 132: an undergraduate course in probabilistic modeling

ECS 145: a course in scripting languages (Python, R)

ECS 158: an undergraduate course in parallel computation

ECS 256: a graduate course in probabilistic modeling

ridgelm,plot.rlm *Ridge Regression*

Description

Similar to `lm.ridge` in `MASS` packaged included with R, but with a different kind of scaling and a little nicer plotting.

Usage

```
ridgelm(xy,lambda = seq(0.01,1,0.01),mapback=TRUE)
## S3 method for class 'rlm'
plot(x,y,...)
```

Arguments

`xy`	Data, response variable in the last column.
`lambda`	Vector of desired values for the ridge parameter.
`mapback`	If TRUE, the scaling that had been applied to the original data will be map back to the original scale, so that the estimated regression coefficients are now on the scale of the original data.
`x`	Object of type 'rlm', output of `ridgelm`.
`y`	Needed for consistency with the generic. Not used.
`...`	Needed for consistency with the generic. Not used.

Details

Centers and scales the predictors X, and centers the response variable Y. Computes X'X and then solves [(X'X)/n + lambda I]b = X'Y/n for b. The 1/n factors are important, making the diagonal elements of (X'X)/n all 1s and thus facilitating choices for the lambdas in a manner independent of the data.

Calling `plot` on the output of `ridgelm` dispatches to `plot.rlm`, thus diplaying the ridge traces.

Value

The function `ridgelm` returns an object of class 'rlm', with components `bhats`, the estimated beta vectors, one column per lambda value, and `lambda`, a copy of the input.

Author(s)

Norm Matloff

TStoX *Transform Time Series to Rectangular Form*

Description

Inputs a time series and transforms it to a form suitable for prediction using `lm` etc.

Usage

```
TStoX(x,lg,y=NULL)
TStoMat(xmat,lg,y)
```

Arguments

`x`	A vector.
`lg`	Lag, a positive integer.
`xmat`	A matrix, data frame etc., with each column a time series, over a common time period.
`y`	A time series, again on that common time period If NULL, it is set to `x`.

Details

`TStoX` is for transforming vectors, while `TStoMat` handles the multivariate time series case. Intended for use with `lm` or other regression model, predicting `y[i]` from observations `i-lg,i-lg+1,...,i-1`.

Value

Let m denote length of x, and in the matrix input case, the number of rows in xmat. Let p be 1 in the vector case, ncol(xmat) in the matrix case. The return value is a matrix with m-lg rows. There will be p*lg+1 columns, with "Y," the numbers to be predicted in the last column. y[lg+1],y[lg+2],...,y[m].

In the matrix case, in a given row, there will be all lg recent observations for the first time series, then all lg recent observations for the second one, and so on, and finally the y value.

Author(s)

Norm Matloff

Examples

```
set.seed(9999)
z <- sample(1:100,12)
z
# [1] 87 66 79 21 67 81 97 77 92 68 74  3
TStoX(z,3)
#      [,1] [,2] [,3] [,4]
# [1,]   87   66   79   21
# [2,]   66   79   21   67
# [3,]   79   21   67   81
# [4,]   21   67   81   97
# [5,]   67   81   97   77
# [6,]   81   97   77   92
# [7,]   97   77   92   68
# [8,]   77   92   68   74
# [9,]   92   68   74    3
set.seed(9999)
zm <- matrix(sample(1:100,24),nrow=2,byrow=TRUE)
y <- sample(1:5,12,replace=TRUE)
zm
#      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
# [1,]   87   66   79   21   67   81   97   77   92    68
# [2,]   15   96   37   80   78    7   69   12   27    84
#      [,11] [,12]
# [1,]    74     3
# [2,]   100    43
y
# [1] 2 2 2 1 3 2 2 4 3 1 5 1
xy <- TStoMat(zm,5,y)
lm(xy[,11] ~ xy[,1:10])  # toy example, need larger m
```

---

unscale *Miscellaneous Utilities*

Description

Utilities.

Usage

```
unscale(scaledx,ctrs=NULL,sds=NULL)
```

Arguments

scaledx	Matrix.
ctrs	Take the original means to be ctrs
sds	Take the original standard deviations to be sds

Value

The function unscale returns the original object to which scale had been applied. Or, the attributes ctrs and sds can be specified by the user.

Author(s)

Norm Matloff

Index