Package 'ma'

June 6, 2017

Type Package

Title Model Averaging

Version 1.0-8

Date 2017-06-06

Imports doParallel, foreach, parallel, quadprog

Suggests crs, np, rgl, knitr, rmarkdown

VignetteBuilder knitr

Depends R (>= 2.10)

Author Jeffrey S. Racine [aut, cre]

Maintainer Jeffrey S. Racine <racinej@mcmaster.ca>

Description Model averaging using a variety of multivariate bases and averaging criteria.

License GPL-2

URL https://github.com/JeffreyRacine/R-Package-ma

BugReports https://github.com/JeffreyRacine/R-Package-ma/issues

NeedsCompilation yes

Archs x86_64

R topics documented:

ma-package *Model Averaging*

Description

Model averaging using a variety of multivariate bases and averaging criteria.

Details

The DESCRIPTION file:

Package:	ma
Type:	Package
Title:	Model Averaging
Version:	1.0-8
Date:	2017-06-06
Imports:	doParallel, foreach, parallel, quadprog
Suggests:	crs, np, rgl, knitr, rmarkdown
VignetteBuilder:	knitr
Depends:	R (>= 2.10)
Authors@R:	person(given = "Jeffrey S.", family = "Racine", role = c("aut","cre"), email = "racinej@mcmaster.ca")
Author:	Jeffrey S. Racine [aut, cre]
Maintainer:	Jeffrey S. Racine <racinej@mcmaster.ca>
Description:	Model averaging using a variety of multivariate bases and averaging criteria.
License:	GPL-2
URL:	https://github.com/JeffreyRacine/R-Package-ma
BugReports:	https://github.com/JeffreyRacine/R-Package-ma/issues

Index of help topics:

```
cps71                   Canadian High School Graduate Earnings
india                   Childhood Malnutrition in India
lm.ma                   Fitting Model Average Models
ma-package              Model Averaging
oecdpanel               Cross Country Growth Panel
plot.lm.ma              Plot an 'lm.ma' Object
wage1                   Cross-Sectional Data on Wages
```

Author(s)

Jeffrey S. Racine [aut, cre]

Maintainer: Jeffrey S. Racine <racinej@mcmaster.ca>

cps71 *Canadian High School Graduate Earnings*

Description

Canadian cross-section wage data consisting of a random sample taken from the 1971 Canadian Census Public Use Tapes for male individuals having common education (grade 13). There are 205 observations in total.

Usage

```
data("cps71")
```

Format

A data frame with 2 columns, and 205 rows.

logwage the first column, of type `numeric`

age the second column, of type `integer`

Source

Aman Ullah

References

Pagan, A. and A. Ullah (1999), *Nonparametric Econometrics,* Cambridge University Press.

Examples

```
## Not run:
data(cps71)
model <- lm.ma(logwage~age,compute.anova=TRUE,data=cps71)
summary(model)
plot(model,plot.data=TRUE,plot.ci=TRUE,plot.B=999)

## End(Not run)
```

india *Childhood Malnutrition in India*

Description

Demographic and Health Survey data on childhood nutrition in India.

Usage

```
data(india)
```

Format

A data frame with 37623 observations on the following 21 variables.

`cheight` child's height (centimeters); a numeric vector

`cage` child's age (months); a numeric vector

`breastfeeding` duration of breastfeeding (months); a numeric vector

`csex` child's sex; a factor with levels `male` `female`

`ctwin` whether or not child is a twin; a factor with levels `single birth` `twin`

`cbirthorder` birth order of the child; a factor with levels `1` `2` `3` `4` `5`

`mbmi` mother's BMI (kilograms per meter squared); a numeric vector

`mage` mother's age (years); a numeric vector

`medu` mother's years of education; a numeric vector

`edupartner` father's years of education; a numeric vector

`munemployed` mother's employment status; a factor variable with levels `unemployed` `employed`

`mreligion` mother's religion; a factor variable with levels `christian` `hindu` `muslim` `other` `sikh`

`mresidence` mother's residential classification; a factor with levels `urban` `rural`

`wealth` mother's relative wealth; a factor with levels `poorest` `poorer` `middle` `richer` `richest`

`electricity` electricity access; a factor with levels `no` `yes`

`radio` radio ownership; a factor with levels `no` `yes`

`television` television ownership; a factor with levels `no` `yes`

`refrigerator` refrigerator ownership; a factor with levels `no` `yes`

`bicycle` bicycle ownership; a factor with levels `no` `yes`

`motorcycle` motorcycle ownership; a factor with levels `no` `yes`

`car` car ownership; a factor with levels `no` `yes`

Source

http://www.econ.uiuc.edu/~roger/research/bandaids/india.Rda

References

Koenker, R. (2011), "Additive models for quantile regression: Model selection and confidence bandaids," Brazilian Journal of Probability and Statistics 25(3), pp. 239-262.

Examples

```
## Not run:
data(india)
attach(india)

faccsex <- factor(csex)
facctwin <- factor(ctwin)
faccbirthorder <- factor(cbirthorder)
facmunemployed <- factor(munemployed)
facmreligion <- factor(mreligion)
faccar <- factor(car)

## Estimate a semiparametric additive model averaged model

model <- lm.ma(cheight ~ faccsex + facctwin + faccbirthorder +
               facmunemployed + facmreligion + faccar + cage +
               mbmi + medu,
               basis="additive",
               vc=FALSE)

summary(model)

plot(model,plot.data=TRUE)
plot(model,plot.deriv=TRUE)

## End(Not run)
```

`lm.ma` *Fitting Model Average Models*

Description

A function with an interface similar to `lm` that averages over a set of linear (in parameters) candidate models.

Usage

```
lm.ma(...)

## Default S3 method:
lm.ma(y = NULL,
      X = NULL,
      X.eval = NULL,
      all.combinations = TRUE,
```

```
      alpha = 0.05,
      auto.basis = c("tensor","taylor","additive"),
      auto.reduce = TRUE,
      B = 99,
      basis.vec = NULL,
      basis = c("auto","tensor","taylor","additive"),
      boot.ci = FALSE,
      compute.anova = FALSE,
      compute.anova.boot = FALSE,
      compute.anova.index = NULL,
      compute.deriv = FALSE,
      compute.mean = TRUE,
      degree.by = 2,
      degree.max = NULL,
      degree.min = 0,
      deriv.index = NULL,
      deriv.order = 1,
      DKL.mat = NULL,
      eps.lambda = 1e-04,
      knots = FALSE,
      lambda.S = 2,
      lambda.max = 1,
      lambda.num.max = NULL,
      ma.weights = NULL,
      ma.weights.cutoff = 1e-04,
      max.dim.candidate.models = 5000,
      max.num.candidate.models = 2500,
      method = c("jma","mma"),
      parallel = FALSE,
      parallel.cores = NULL,
      rank.vec = NULL,
      restrict.sum.ma.weights = TRUE,
      rng.seed = 42,
      S = 1,
      segments.by = 2,
      segments.max = 3,
      segments.min = 1,
      singular.ok = TRUE,
      trace = FALSE,
      vc = TRUE,
      verbose = TRUE,
      weights = NULL,
      ...)

## S3 method for class 'formula'
lm.ma(formula,
      data = list(),
      y = NULL,
```

```
X = NULL,
X.eval = NULL,
all.combinations = TRUE,
alpha = 0.05,
auto.basis = c("tensor","taylor","additive"),
auto.reduce = TRUE,
B = 99,
basis.vec = NULL,
basis = c("auto","tensor","taylor","additive"),
boot.ci = FALSE,
compute.anova = FALSE,
compute.anova.boot = FALSE,
compute.anova.index = NULL,
compute.deriv = FALSE,
compute.mean = TRUE,
degree.by = 2,
degree.max = NULL,
degree.min = 0,
deriv.index = NULL,
deriv.order = 1,
DKL.mat = NULL,
eps.lambda = 1e-04,
knots = FALSE,
lambda.S = 2,
lambda.max = 1,
lambda.num.max = NULL,
ma.weights = NULL,
ma.weights.cutoff = 1e-04,
max.dim.candidate.models = 5000,
max.num.candidate.models = 2500,
method = c("jma","mma"),
parallel = FALSE,
parallel.cores = NULL,
rank.vec = NULL,
restrict.sum.ma.weights = TRUE,
rng.seed = 42,
S = 1,
segments.by = 2,
segments.max = 3,
segments.min = 1,
singular.ok = TRUE,
trace = FALSE,
vc = TRUE,
verbose = TRUE,
weights = NULL,
...)
```

Arguments

`formula` a symbolic description of the model to be fit

`data` an optional data frame containing the variables in the model

`y` a one dimensional vector of dependent data

`X` a p-variate data frame of explanatory (training) data

`X.eval` a p-variate data frame of points on which the regression will be estimated (evaluation data)

`all.combinations` a logical value indicating whether or not to attempt all combinations of degrees, segments, knots, and lambda values (if `all.combinations=FALSE` only candidate models with the same degree in all dimensions are considered, no interior knots are considered, while the minimum number of segments and smallest value for lambda are used)

`alpha` a value in (0,1) used to compute $1-\alpha\%$ confidence intervals

`auto.basis` which (subset possible) bases to use when `basis="auto"`

`auto.reduce` a logical value indicating whether or not to use some crude heuristics to reduce the number of candidate models if the number of candidate models exceeds `max.num.candidate.models`

`B` the number of bootstrap replications desired

`basis.vec` a vector (character) of bases for each candidate model

`basis` a character string indicating whether the generalized Taylor polynomial, additive or tensor product basis should be used (if `basis="auto"` then for each candidate model the most appropriate basis is determined via cross-validation)

`boot.ci` a logical value indicating whether or not to construct nonparametric bootstrap confidence intervals

`compute.anova` a logical value indicating whether or not to conduct an anova-based procedure to test for predictor significance

`compute.anova.boot` a logical value indicating whether or not the test for predictor significance uses asymptotic or bootstrapped P-values

`compute.anova.index` an optional vector of indices indicating which predictor(s) are to be tested (default is all predictors)

`compute.deriv` a logical value indicating whether or not to compute derivatives

`compute.mean` a logical value indicating whether or not to compute the conditional mean

`degree.by` increment in degree sequence (if `degree.min=0` sequence will include degree 0 then start at 1 in increments of `degree.by`)

`degree.max` the maximum value for the basis degree in each dimension (the value defaults to `max(2,ceiling(log(n)-S*log(1+k)))` where `k` is the number of numeric predictors and `n` the number of observations

`degree.min` the minimum value for the basis degree in each dimension deriv.index — an optional vector of indices indicating which predictor(s) derivative is computed

deriv.order — an integer indicating the order of derivative desired (1,2,...)

DKL.mat — a matrix with degree, knots, and lambda values (if `vc=TRUE`) that could optionally be passed to the basis routines

eps.lambda — a small positive constant for the start of the sequence of smoothing parameters used in the weight function for the categorical predictors when `vc=TRUE`

knots — a logical value indicating whether or not to include interior knots

lambda.S — the constant in the data-driven rule for determining `lambda.num.max`

lambda.max — largest value (<= 1) of the smoothing parameters used in the weight function for the categorical predictors when `vc=TRUE`

lambda.num.max — the maximum value for the smoothing parameter grid in each dimension (defaults to `max(2,ceiling(log(n)-lambda.S*log(1+p)))` where `p` is the number of categorical predictors and `n` the number of observations

ma.weights — a vector of model average weights obtained from a previous invocation (useful for bootstrapping etc.)

ma.weights.cutoff — a small number below which a model weight is deemed to be essentially zero

max.dim.candidate.models — an arbitrary upper bound on the maximum dimension of candidate models permitted

max.num.candidate.models — an arbitrary upper bound on the maximum number of candidate models permitted

method — a character string indicating whether to use jackknife model averaging (`"jma"`, Hansen and Racine (2013)) or Mallows model averaging (`"mma"`, Hansen (2007) - both are frequentist model average criterion)

parallel — a logical value indicating whether or not to run certain routines in parallel

parallel.cores — a positive integer indicating the number of cores desired when `parallel=TRUE` (when `parallel=FALSE` defaults to the number of available cores)

rank.vec — a vector of ranks for each candidate model

restrict.sum.ma.weights — a logical value indicating whether or not to restrict the sum of the model average weights to one when solving the quadratic program (they are normalized afterwards when `restrict.sum.ma.weights=FALSE`)

rng.seed — an integer used to seed R's random number generator - this is to ensure replicability when bootstrapping

S — the constant in the data-driven rule for determining `degree.max`

segments.by — increment in segments sequence when `knots=TRUE`

segments.min — the minimum number of segments when `knots=TRUE` (i.e., number of knots minus 1 - there always exist two knots, the endpoints) to allow for the B-spline basis segments.max — the maximum number of segments when knots=TRUE (i.e., number of knots minus 1 - there always exist two knots, the endpoints) to allow for the B-spline basis

singular.ok — if 'FALSE' (the default in S but not in R) a singular fit is an error

trace — a logical value indicating whether or not to issue a detailed progress report via warning

vc — a logical value indicating whether to allow the categorical predictors to enter additively (only the intercept can shift) or to instead use a varying coefficient structure (all parameters can shift)

verbose — a logical value indicating whether to report detailed progress during computation (warnings() are not issued when verbose=FALSE)

weights — an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector; if non-NULL, weighted least squares is used with weights weights (that is, minimizing $\sum_i w_i e_i^2$); otherwise ordinary least squares is used

... — optional arguments to be passed

Details

Models for lm.ma are specified symbolically. A typical model has the form `response ~ terms` where response is the (numeric) response vector and terms is a series of terms which specifies a linear predictor for response. Typical usages are

```
model <- lm.ma(y~x1+x2)
model <- lm.ma(y~x1+x2,compute.deriv=TRUE)
model <- lm.ma(y~x1+x2,boot.ci=TRUE)
model <- lm.ma(y~x1+x2,compute.anova=TRUE,compute.anova.boot=TRUE,degree.min=1)
model <- lm.ma(y~x1+x2,parallel=TRUE)
model <- lm.ma(y~x1+x2,parallel=TRUE,parallel.cores=2)
model <- lm.ma(y~x+z,lambda.S=3)
model <- lm.ma(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,
               vc=FALSE,
               degree.by=1,
               degree.max=5,
               basis="additive",
               all.combinations=FALSE)

plot(model)
plot(model,plot.data=TRUE)
plot(model,plot.ci=TRUE,plot.B=199)
plot(model,plot.data=TRUE,plot.ci=TRUE,plot.B=199)
plot(model,plot.deriv=TRUE)
plot(model,plot.deriv=TRUE,plot.ci=TRUE,plot.B=399)

summary(model)
fitted(model)
coef(model)
```

```
## For generating predictions, create foo, a data frame with named
## elements (important) for all predictors in the object model,
## then call predict, e.g.,

foo <- data.frame(x1=c(1,2),x2=c(3,4))
predict(model,newdata=foo)

## If you want to see the degrees, number of segments, and smoothing
## parameters for the categorical predictors (vc=TRUE) selected by
## the procedure for the models that receive positive model average
## weights, try the following:

model$DKL.mat[model$ma.weights>model$ma.weights.cutoff,]
```

Note that, unlike `lm` in which the formula interface specifies functional form, in `lm.ma` the formula interface is strictly for listing the variables involved and the procedure will determine an appropriate model averaged functional form. Do not incorporate transformations, interactions and the like in the formula interface for `lm.ma` as these will most surely fail.

This function computes a model that is the weighted average of a set of least squares candidate models whose predictors are generated by common basis functions (additive, generalized Taylor polynomial, or tensor products). The candidate models increase in complexity from linear bases (if `degree.min=1`) through higher order ones up to `degree.max`. All bases are of the Bernstein polynomial class, as opposed to raw polynomials, and allow for differing degrees across multivariate predictors. When `knots=TRUE`, interior knots are used and the Bernstein polynomials become B-spline bases and we are then averaging over regression spline models. When the number of numeric predictors is two or more, the generalized Taylor polynomial includes interaction terms up to order degree minus one. Since we are averaging over models that are nonlinear in the predictors, derivatives will be vectors that potentially depend on the values of every predictor. An ad-hoc formula is used to determine the relationship between the largest (most complex) model, the sample size, and the number of predictors. This ad-hoc rule was set so that, as the sample size increases, we can approximate ever more complex functions while necessarily restricting the size of the largest model in small sample settings. Categorical predictors can enter additively and linearly (if `vc=FALSE`) or in a parsimonious manner by exploiting recent developments in semiparametric varying coefficient models along the lines of Li, Ouyang, and Racine (2013). With the options `knots=TRUE` and `vc=TRUE`, we are averaging over varying-coefficient regression splines.

This approach frees the user from using either model assertion or selection methods and thereby attenuates bias arising from model misspecification. Simulations reveal that this approach is competitive with some semi- and nonparametric approaches. Because it uses only least squares fits, it can be more computationally efficient than its nonparametric counterparts.

Value

`lm.ma` returns an object of class "lm.ma".

The function summary is used to obtain and print a summary of the results. The generic accessor functions `coef`, `fitted`, `predict`, `plot` (see `?plot.lm` for details) and `residuals` extract various useful features of the value returned by lm.ma.

An object of class "lm.ma" is a list containing at least the following components:

`degree.max`	value of degree.max for each dimension (set by an ad-hoc rule unless manually overridden)
`deriv.ci.l`	$\alpha/2$ nonparametric confidence value matrix for the matrix of derivatives
`deriv.ci.u`	$1-\alpha/2$ nonparametric confidence value matrix for the matrix of derivatives
`deriv.scale`	robust scale (mad) matrix for the matrix of derivatives
`deriv`	matrix of derivative vectors for each predictor
`fitted.ci.l`	$\alpha/2$ nonparametric confidence value vector for the vector of fitted/predicted values
`fitted.ci.u`	$1-\alpha/2$ nonparametric confidence value vector for the vector fitted/predicted values
`fitted.scale`	robust scale (mad) vector for the vector of fitted/predicted values
`fitted.values`	vector of fitted values
`ma.weights`	model average weights
`r.squared`	appropriate measure of goodness of fit (Doksum and Samarov (1995))
`residuals`	model residuals

Note

This code is in beta status until further notice - proceed accordingly.

Note that the purpose of this package is to attenuate bias arising from model misspecification in situations where model uncertainty is present and you are concerned about its impact on any subsequent inference and prediction. This package is best suited to situations involving a manageable number of predictors (i.e., a handful or two at most) and a sufficient number of observations so that nonlinearities can reasonably be uncovered. If your objective is to include all possible measured predictors (i.e., the kitchen sink approach) and conduct variable selection (i.e., attempt to determine which variables enter linearly), this package is not for you; see instead the R packages **BMA**, **lars**, or the function stepAIC in the **MASS** package (with `degree.max=1` the defaults would only allow for at most eleven numeric predictors, i.e., 2^{11} combinations of degrees 0 and 1). To get around this limitation that arises by attempting to consider a range of degree, segment, and smoothing parameter values for each dimension (the number of combinations can quickly get far too large), the option `all.combinations=FALSE` can be invoked. This restricts the number of candidate models by holding the degree, segment, and smoothing parameters to be the same for each dimension which can reduce the number of models to just a handful at most. Using `basis="additive"` further restricts the rank of each candidate model, while `vc=FALSE` can reduce execution time in the presence of categorical predictors (see the example in **Details** above).

The number of candidate models may grow unreasonably large (say 2,500 or more) if multiple predictors are present. Some heuristics are therefore necessary in order to corral the number of candidate models (and the maximum basis dimension). However, no default setting can be ideal for all data generating processes and you may wish to intervene. If you wish to reduce the number of candidate models used, there are a number of ways of accomplishing this. In particular, you might want to i) increase `S`, ii) increase `lambda.S` (if categorical predictors are present and `vc=TRUE`), iii) set and restrict `degree.max`, iv) set and restrict `lambda.num.max` if categorical predictors are present, v) reduce `segments.max` (if `knots=TRUE`), vi) set `all.combinations=FALSE`, vi) directly modify `max.dim.candidate.models` and/or `max.num.candidate.models`, or perhaps instead consider a semiparametric model (`basis="additive"` and `vc=FALSE` produces semiparametric additive candidate models - see the example in `?india` for an illustration). When building the final model each candidate model must be constructed and evaluated. However, after solving for the model average weights, a number of candidate models may be assigned essentially zero weight. Subsequently, only the non-zero weight models need be evaluated (e.g. when constructing derivative estimates, predictions, confidence intervals and the like).

When `compute.anova.boot=TRUE`, the option `compute.anova` uses a bootstrap procedure that requires re-computation of the model average model for each bootstrap replication. With one or two predictors and `compute.anova.boot=TRUE` the procedure may be fairly fast, but as the model complexity increases the procedure will require some patience.

The option `compute.anova=TRUE` cannot be used in the presence of one or more factors and exactly one numeric predictor since there is no numeric predictor present when testing for significance for the one numeric predictor.

Note that the option `compute.anova=TRUE` (not default) will warn immediately when `degree.min=0` (default) and rest to `degree.min=1`. The reason for this is because irrelevant predictors can be automatically removed without the need for pre-testing if the procedure selects the degree for any predictor to be 0 - in such cases the restricted and unrestricted models may coincide and the test is degenerate. The same holds for smoothing parameter values with `vc=TRUE` in the presence of categorical predictors (when `lambda=1` irrelevant categorical predictors are automatically removed without the need for pretesting, so we need to rule this case out when conducting hypothesis tests).

Averaging over models with ill-conditioned bases is not advised. Pay attention to the warning "Dimension basis is ill-conditioned - reduce `degree.max`" should it arise and reduce `degree.max` until this no longer is the case.

If you wish to plot the object with the option `plot.ci=TRUE`, it is not necessary to use the option `boot.ci=TRUE` in the call to `lm.ma()` (this will simply add to overhead)

Note that `predict.ma` produces a vector of predictions or a list of predictions, confidence bounds, derivative matrices and their confidence bounds.

See the examples contained in `demo(package="ma")` for illustrative demonstrations with real and simulated data (e.g., `demo(cps71,package="ma")`).

Author(s)

Jeffrey S. Racine

References

Doksum, K. and A. Samarov (1995), "Nonparametric Estimation of Global Functionals and a Measure of the Explanatory Power of Covariates in Regression," The Annals of Statistics, 23 1443-1473.

Li, Q. and D. Ouyang and J.S. Racine (2013), "Categorical Semiparametric Varying Coefficient Models," Journal of Applied Econometrics, Volume 28, 551-579.

Hansen, B. E. (2007), "Least Squares Model Averaging," Econometrica 75, 1175-1189.

Hansen, B. E. & Racine, J. S. (2012), "Jackknife Model Averaging," Journal of Econometrics 167(1), 38-46.

Racine, J.S. and D. Zhang and Q. Li (2017), "Model Averaged Categorical Regression Splines."

See Also

lm, crs, npreg

Examples

```
options(warn=-1)
#### Example 1 - simulated nonlinear one-predictor function

set.seed(42)
n <- 100
x <- sort(runif(n))
dgp <- cos(2*pi*x)
y <- dgp + rnorm(n,sd=0.5*sd(dgp))

model.ma <- lm.ma(y~x)

summary(model.ma)

## Note that the following calls to plot() use the option
## plot.ci=TRUE which then invokes a bootstrap procedure. The
## plots may take a few seconds to appear due to this additional
## computation (if you remove this option the plots will appear
## sooner).

par(mfrow=c(1,2))
plot(model.ma,plot.data=TRUE,plot.ci=TRUE)
plot(model.ma,plot.data=TRUE,plot.ci=TRUE,plot.deriv=TRUE)
par(mfrow=c(1,1))

#### Example 2 - five predictor (two categorical) earnings function
data(wage1)
attach(wage1)

## Classical linear regression model (linear, additive, no interactions)

model.lm <- lm(lwage ~ female + married + educ + exper + tenure)

## Murphy-Welch's favourite specification
model.lm.mw <- lm(lwage ~ female + married + educ + exper + I(exper^2)
                  + I(exper^3) + I(exper^4) + tenure)

## Murphy-Welch's favourite specification with interactions in the intercepts
model.lm.mwint <- lm(lwage ~ female + married + female:married + educ + exper
                     + I(exper^2) + I(exper^3) + I(exper^4) + tenure)

summary(model.lm)

## Compare with a semiparametric additive model average estimator
## (female and married are factors)

model.ma <- lm.ma(lwage ~ female + married + educ + exper + tenure,
                  compute.deriv = TRUE,
```

```
                  basis = "additive",
                  degree.by = 1,
                  vc = FALSE)

summary(model.ma)

## Compare coefficients from the simple linear model with the (vector summary) values
## from model averaging for the non-factor predictors

apply(coef(model.ma),2,summary)
coef(model.lm)[4:6]

## Compute parametric and model averaged marriage premiums for males and females
## at median values of remaining predictors

newdata.female.married <- data.frame(educ=round(median(educ)),
                                     exper=round(median(exper)),
                                     tenure=round(median(tenure)),
                                     female=factor("Female",levels=levels(female)),
                                     married=factor("Married",levels=levels(married)))

newdata.female.notmarried <- data.frame(educ=round(median(educ)),
                                        exper=round(median(exper)),
                                        tenure=round(median(tenure)),
                                        female=factor("Female",levels=levels(female)),
                                    married=factor("Notmarried",levels=levels(married)))

## Compute the so-called marriage premium - try three simple parametric
## specifications (take your pick - is the premium +13%? +3%? -12%?)

## Linear parametric
predict(model.lm,newdata=newdata.female.married)-
predict(model.lm,newdata=newdata.female.notmarried)

## Murphy-Welch parametric
predict(model.lm.mw,newdata=newdata.female.married)-
predict(model.lm.mw,newdata=newdata.female.notmarried)

## Murphy-Welch parametric augmented with a dummy interaction
predict(model.lm.mwint,newdata=newdata.female.married)-
predict(model.lm.mwint,newdata=newdata.female.notmarried)

## Model average
predict(model.ma,newdata=newdata.female.married)$fit-
predict(model.ma,newdata=newdata.female.notmarried)$fit

detach(wage1)

#### Example 3 - Canadian Current Population Survey earnings data

## We compute two nonparametric estimators to compare with the
## model averaging approach.
```

```
suppressPackageStartupMessages(require(np))
suppressPackageStartupMessages(require(crs))
data(cps71)
attach(cps71)
model.ma <- lm.ma(logwage~age)
plot(model.ma,plot.data=TRUE)
model.kernel <- npreg(logwage~age,regtype="ll",bwmethod="cv.aic")
lines(age,fitted(model.kernel),col=4,lty=4,lwd=2)
model.spline <- crs(logwage~age,cv.threshold=0)
lines(age,fitted(model.spline),col=3,lty=3,lwd=2)
legend("topleft",c("Model Average",
                   "Nonparametric Kernel",
                   "Nonparametric B-Spline"),
                   col=c(1,4,3),
                   lty=c(1,4,3),
                   lwd=c(1,2,2),
                   bty="n")

summary(model.spline)
summary(model.kernel)
summary(model.ma)

detach(cps71)

#### Example 5 - simulated multiplicative nonlinear two-predictor function

suppressPackageStartupMessages(require(rgl))

set.seed(42)
n <- 1000
x1 <- runif(n)
x2 <- runif(n)

dgp <- cos(2*pi*x1)*sin(2*pi*x2)

y <- dgp + rnorm(n,sd=0.5*sd(dgp))

n.eval <- 25
x.seq <- seq(0,1,length=n.eval)
newdata <- data.frame(expand.grid(x.seq,x.seq))
names(newdata) <- c("x1","x2")

model.ma <- lm.ma(y~x1+x2)

summary(model.ma)

## Use the rgl package to render a 3D object (RGL is a 3D real-time rendering
## system for R that supports OpenGL, among other formats).

z <- matrix(predict(model.ma,newdata=newdata),n.eval,n.eval)
num.colors <- 1000
colorlut <- topo.colors(num.colors)
col <- colorlut[ (num.colors-1)*(z-min(z))/(max(z)-min(z)) + 1 ]
```

```
par(ask=TRUE)
readline(prompt = "Hit <Return> to see next plot:")
open3d()
par3d(windowRect=c(900,100,900+640,100+640))
rgl.viewpoint(theta = 0, phi = -70, fov = 80)
persp3d(x.seq,x.seq,z=z,
        xlab="X1",ylab="X2",zlab="Y",
        ticktype="detailed",
        border="red",
        color=col,
        alpha=.7,
        back="lines",
        main="Conditional Mean")
grid3d(c("x", "y+", "z"))

## Note - if you click on the rgl window you can rotate the estimate
## by dragging the object, zoom in and out etc.
```

---

oecdpanel *Cross Country Growth Panel*

Description

Cross country GDP growth panel covering the period 1960-1995 used by Liu and Stengos (2000) and Maasoumi, Racine, and Stengos (2007). There are 616 observations in total.

Usage

```
data("oecdpanel")
```

Format

A data frame with 7 columns, and 616 rows. This panel covers 7 5-year periods: 1960-1964, 1965-1969, 1970-1974, 1975-1979, 1980-1984, 1985-1989 and 1990-1994.

A separate local-linear `rbandwidth` object ('bw') has been computed for the user's convenience which can be used to visualize this dataset using `plot(bw)`.

growth the first column, of type `numeric`: growth rate of real GDP per capita for each 5-year period

oecd the second column, of type `factor`: equal to 1 for OECD members, 0 otherwise

year the third column, of type `integer`

initgdp the fourth column, of type `numeric`: per capita real GDP at the beginning of each 5-year period

popgro the fifth column, of type `numeric`: average annual population growth rate for each 5-year period

inv the sixth column, of type `numeric`: average investment/GDP ratio for each 5-year period

humancap the seventh column, of type `numeric`: average secondary school enrolment rate for each 5-year period

Source

Thanasis Stengos

References

Liu, Z. and T. Stengos (1999), “Non-linearities in cross country growth regressions: a semiparametric approach,” Journal of Applied Econometrics, 14, 527-538.

Maasoumi, E. and J.S. Racine and T. Stengos (2007), “Growth and convergence: a profile of distribution dynamics and mobility,” Journal of Econometrics, 136, 483-508

Examples

```
## Not run:
data(oecdpanel)
attach(oecdpanel)
oecd <- factor(oecd)
year <- ordered(year)
model.ma <- lm.ma(growth ~ oecd + year + initgdp + popgro + inv + humancap)
summary(model.ma)
plot(model.ma,plot.data=TRUE,plot.rug=TRUE)

## End(Not run)
```

---

plot.lm.ma   *Plot an* `lm.ma` *Object*

Description

Plots a model average model and its derivatives.

Usage

```
## S3 method for class 'lm.ma'
plot(x,
     plot.B = 99,
     plot.ci = FALSE,
     plot.data = FALSE,
     plot.deriv = FALSE,
     plot.num.eval = 250,
     plot.rug = FALSE,
     plot.xtrim = 0.005,
     ...)
```

Arguments

`x`	an object of type `lm.ma`
`plot.B`	number of bootstrap replications used to construct nonparametric confidence intervals
`plot.ci`	a logical value indicating whether to plot nonparametric confidence intervals or not
`plot.data`	a logical value indicating whether to plot the data or not
`plot.deriv`	a logical value indicating whether to compute derivatives or not
`plot.num.eval`	number of evaluation points
`plot.rug`	a logical value indicating whether to plot the data with a rug or not
`plot.xtrim`	trimming parameter used to exclude tail values for the predictors that can obscure main features in the plot (trims the proportion `plot.xtrim` from each tail)
`...`	optional arguments to be passed to `plot`

Details

This function plots an object returned by `lm.ma`. Typical usages are

```
plot(model)
plot(model,plot.data=TRUE)
plot(model,plot.ci=TRUE,plot.B=99)
plot(model,plot.data=TRUE,plot.ci=TRUE,plot.B=199)
plot(model,plot.deriv=TRUE)
plot(model,plot.deriv=TRUE,plot.ci=TRUE,plot.B=399)
```

Value

None.

Author(s)

Jeffrey S. Racine

References

Racine, J.S. and D. Zhang and Q. Li (2017), “Model Averaged Categorical Regression Splines.”

Examples

```
data(cps71)
model <- lm.ma(logwage~age,data=cps71)
plot(model,plot.data=TRUE,plot.ci=TRUE)
```

`wage1` *Cross-Sectional Data on Wages*

Description

Cross-section wage data consisting of a random sample taken from the U.S. Current Population Survey for the year 1976. There are 526 observations in total.

Usage

```
data("wage1")
```

Format

A data frame with 24 columns, and 526 rows.

wage column 1, of type `numeric`, average hourly earnings

educ column 2, of type `numeric`, years of education

exper column 3, of type `numeric`, years potential experience

tenure column 4, of type `numeric`, years with current employer

nonwhite column 5, of type `factor`, =“Nonwhite” if nonwhite, “White” otherwise

female column 6, of type `factor`, =“Female” if female, “Male” otherwise

married column 7, of type `factor`, =“Married” if Married, “Nonmarried” otherwise

numdep column 8, of type `numeric`, number of dependents

smsa column 9, of type `numeric`, =1 if live in SMSA

northcen column 10, of type `numeric`, =1 if live in north central U.S

south column 11, of type `numeric`, =1 if live in southern region

west column 12, of type `numeric`, =1 if live in western region

construc column 13, of type `numeric`, =1 if work in construc. indus.

ndurman column 14, of type `numeric`, =1 if in nondur. manuf. indus.

trcommpu column 15, of type `numeric`, =1 if in trans, commun, pub ut

trade column 16, of type `numeric`, =1 if in wholesale or retail

services column 17, of type `numeric`, =1 if in services indus.

profserv column 18, of type `numeric`, =1 if in prof. serv. indus.

profocc column 19, of type `numeric`, =1 if in profess. occupation

clerocc column 20, of type `numeric`, =1 if in clerical occupation

servocc column 21, of type `numeric`, =1 if in service occupation

lwage column 22, of type `numeric`, log(wage)

expersq column 23, of type `numeric`, $exper^2$

tenursq column 24, of type `numeric`, $tenure^2$

Source

Jeffrey M. Wooldridge

References

Wooldridge, J.M. (2000), *Introductory Econometrics: A Modern Approach*, South-Western College Publishing.

Examples

```
## Not run:
data(wage1)
model.ma <- lm.ma(lwage ~ female + married + educ + exper + tenure,
                  data = wage1,
                  parallel = TRUE)
plot(model.ma,plot.data=TRUE)
plot(model.ma,plot.deriv=TRUE)

## End(Not run)
```

Index