Package 'polywog'

May 14, 2025

Title Bootstrapped Basis Regression with Oracle Model Selection

Version 0.4-2

Date 2025-05-13

Description Routines for flexible functional form estimation via basis regression, with model selection via the adaptive LASSO or SCAD to prevent overfitting.

License GPL (>= 2)

URL https://github.com/brentonk/polywog

Depends miscTools (>= 0.6-12)

Imports foreach, Formula, glmnet (>= 1.9-5), iterators, Matrix, ncvreg (>= 2.4-0), Rcpp (>= 0.11.0), stringr

LinkingTo Rcpp

Suggests carData, lattice, rgl

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation yes

Author Brenton Kenkel [aut, cre], Curtis S. Signorino [aut]

Maintainer Brenton Kenkel <brenton.kenkel@gmail.com>

Repository CRAN

Date/Publication 2025-05-13 22:20:02 UTC

Contents

`bootPolywog` *Bootstrap a fitted polywog model*

Description

Nonparametric bootstrap of the `polywog` regression procedure. Can be run on a fitted model of class `"polywog"`, or within the original procedure via the `boot` argument. The function `control.bp` can be used to pass options to `bootPolywog` when bootstrapping within `polywog`.

Usage

```
bootPolywog(
  model,
  nboot = 100,
  .parallel = FALSE,
  reuse.lambda = FALSE,
  reuse.penwt = FALSE,
  nlambda = 100,
  lambda.min.ratio = 1e-04,
  nfolds = 10,
  thresh = NULL,
  maxit = NULL,
  maxtries = 1000,
  min.prop = 0,
  report = FALSE,
  .matrixOnly = FALSE
)

control.bp(
  .parallel = FALSE,
  reuse.lambda = FALSE,
  reuse.penwt = FALSE,
  nlambda = 100,
  lambda.min.ratio = 1e-04,
  nfolds = 10,
  thresh = NULL,
  maxit = NULL,
  maxtries = 1000,
  min.prop = 0,
```

```
  report = FALSE
)
```

Arguments

model	a fitted model of class "polywog", typically the output of polywog or the "polywog.fit" element of the output of cv.polywog.
nboot	number of bootstrap iterations.
.parallel	logical: whether to perform computations in parallel using a backend registered with foreach.
reuse.lambda	logical: whether to use the penalization parameter from the original fit (TRUE), or to cross-validate within each iteration (FALSE, default).
reuse.penwt	logical: whether to use the penalty weights from the original fit (TRUE), or to re-calculate them within each iteration (FALSE, default).
nlambda	number of values of the penalty factor to examine in cross-validation, as in polywog.
lambda.min.ratio	ratio of the smallest value of the penalty factor to the largest, as in polywog.
nfolds	number of cross-validation folds to use.
thresh	convergence threshold, as in polywog. If NULL, use the same value as in the original model.
maxit	iteration limit for fitting, as in polywog. If NULL, use the same value as in the original model.
maxtries	maximum number of attempts to generate a bootstrap sample with a non-collinear model matrix (often problematic with lopsided binary regressors) before stopping and issuing an error message.
min.prop	for models with a binary response variable, minimum proportion of non-modal outcome to ensure is included in each bootstrap iteration (for example, set min.prop = 0.1 to throw out any bootstrap iteration where less than 10 percent or more than 90 percent of the observations are 1's).
report	logical: whether to print a status bar. Not available if .parallel = TRUE.
.matrixOnly	logical: whether to return just the matrix of bootstrap coefficients (TRUE), or the originally supplied model with the bootstrap matrix as the boot.matrix element (FALSE, default).

Details

Parallel computation via the .parallel argument requires registation of a backend for %dopar%, as in polywog. In the case of bootPolywog, bootstrap fitting is carried out in parallel, while cross-validation to choose the penalization factor (assuming reuse.lambda = FALSE) is carried out sequentially within each iteration.

Value

If .matrixOnly = FALSE, the returned object is model with the bootstrap matrix included as its boot.matrix element. If .matrixOnly = TRUE, just the matrix is returned. In either case, the bootstrap matrix is a sparse matrix of class "dgCMatrix-class".

Author(s)

Brenton Kenkel and Curtis S. Signorino

Examples

```
## Using occupational prestige data
data(Prestige, package = "carData")
Prestige <- transform(Prestige, income = income / 1000)

## Fit a polywog model without bootstrapping
## (note: using low convergence threshold to shorten computation time of the
## example, *not* recommended in practice!)
fit1 <- polywog(prestige ~ education + income + type,
                data = Prestige,
                degree = 2,
                thresh = 1e-4)
summary(fit1)

## Bootstrap the fitted model
fit2 <- bootPolywog(fit1, nboot = 5)
summary(fit2)

## Example of parallel processing on Mac/Unix via 'doMC'
## Not run:
library(doMC)
registerDoMC()

fit2 <- bootPolywog(fit1, nboot = 100, .parallel = TRUE)

## End(Not run)

## Example of parallel processing on Windows via 'doSMP'
## Not run:
library(doSMP)
w <- startWorkers()
registerDoSMP(w)

fit2 <- bootPolywog(fit1, nboot = 100, .parallel = TRUE)

stopWorkers(w)

## End(Not run)
```

`cv.polywog` *Cross-validation of polynomial degree and penalization factor*

Description

k-fold cross-validation to select the polynomial degree and penalization factor for a `polywog` model.

Usage

```
cv.polywog(
  formula,
  ...,
  degrees.cv = 1:3,
  nfolds = 10,
  model = TRUE,
  X = FALSE,
  y = FALSE
)
```

Arguments

formula	model formula specifying the response and input variables.
...	other arguments to be passed to polywog. Arguments related to the bootstrap will be ignored, as bootstrapping must be performed separately.
degrees.cv	vector of polynomial degrees to examine via cross-validation.
nfolds	number of folds to use in cross-validation.
model	logical: whether to include the model frame in the "polywog" object included in the output.
X	logical: whether to include the raw model matrix (i.e., the matrix of input variables prior to taking their polynomial expansion) in the "polywog" object included in the output.
y	logical: whether to include the response variable in the "polywog" object included in the output.

Details

When fitting with method = "scad", different fold assignments are used for each polynomial degree specified, because cv.ncvreg does not allow for custom fold assignments. This may affect the accuracy of the estimated cross-validation error for each degree. When method = "scad", the calls to polywog made by cv.polywog will issue warnings that the foldid argument is being ignored.

Value

An object of class "cv.polywog", a list containing:

results A table of each degree tested, the optimal penalization factor λ for that degree, and its cross-validation error.

degree.min The polynomial degree giving the lowest cross-validation error.

polywog.fit A polywog model, fit at the polynomial degree giving the lowest cross-validation error.

Because the returned object contains the fitted polywog model for the optimal degree, no additional runs of polywog are necessary to estimate coefficients or the penalization factor λ. However, bootstrap results must be obtained by running bootPolywog on the "polywog.fit" element of the returned object, as in the examples below.

Author(s)

Brenton Kenkel and Curtis S. Signorino

Examples

```
## Using occupational prestige data
data(Prestige, package = "carData")
Prestige <- transform(Prestige, income = income / 1000)

## Examine degrees 1 through 4
## (note: using low convergence threshold to shorten computation time of the
## example, *not* recommended in practice!)
set.seed(39)
cv1 <- cv.polywog(prestige ~ education + income + type,
                  data = Prestige,
                  degrees.cv = 1:4,
                  nfolds = 10,
                  thresh = 1e-4)
print(cv1)

## Extract best model and bootstrap
fit1 <- cv1$polywog.fit
fit1 <- bootPolywog(fit1, nboot = 5)
summary(fit1)
```

`margEff.polywog` *Marginal effects for polywog models*

Description

Computes average and observationwise marginal effects from a fitted `polywog` model.

Usage

```
## S3 method for class 'polywog'
margEff(object, xvar = NULL, drop = FALSE, ...)
```

Arguments

`object`	a fitted model of class `"polywog"`, typically the output of `polywog`. The object must have a `model` element, meaning it was fit with `model = TRUE`.
`xvar`	a character string containing the name of a raw input variable (from `object$varNames`). Partial matches are allowed.
`drop`	logical: whether to convert one-column matrices in the output to vectors.
`...`	other arguments, currently ignored.

Details

For input variables that are binary, logical, or factors, `margEff.polywog` computes a first difference with comparison to a reference category. All other variables are treated as continuous: the function computes the partial derivative of the fitted value with respect to the selected variable.

Value

If `xvar` is specified, a numeric object containing the marginal effect of the chosen variable at each observation in `object$model`. For factor variables, if there are more than two levels or `drop = FALSE`, the returned object is a matrix; otherwise it is a vector.

If `xvar` is `NULL`, a list of such results for each raw input variable in the model is returned.

In either case, the returned object is of class `"margEff.polywog"`.

Author(s)

Brenton Kenkel and Curtis S. Signorino

See Also

To plot the density of the observationwise marginal effects, see `plot.margEff.polywog`. For a table of average marginal effects and order statistics, `summary.margEff.polywog`.

To compute fitted values, see `predict.polywog` and `predVals`.

Examples

```
## Using occupational prestige data
data(Prestige, package = "carData")
Prestige <- transform(Prestige, income = income / 1000)

## Fit a polywog model
## (note: using low convergence threshold to shorten computation time of the
## example, *not* recommended in practice!)
set.seed(22)
fit1 <- polywog(prestige ~ education + income | type,
                data = Prestige,
                degree = 2,
                thresh = 1e-4)

## Compute marginal effects for all variables
me1 <- margEff(fit1)
summary(me1)  # type was included linearly, hence constant effects

## Plotting density of the results
plot(me1)

## Can do the same when just examining a single variable
me2 <- margEff(fit1, xvar = "income")
summary(me2)
plot(me2)
```

model.frame.polywog *Model frame of a polywog model*

Description

Extracts the model frame from a fitted polywog model, as model.frame.lm does for a fitted lm model.

Usage

```
## S3 method for class 'polywog'
model.frame(formula, ...)
```

Arguments

formula	a fitted model of class "polywog" (the argument is named formula for consistency with the generic function model.frame)
...	other arguments, currently ignored (but may later be adapted for use as in model.frame.lm)

Value

A data frame containing the variables used to fit the model, with additional attributes (e.g., "terms") used to construct a model matrix.

Author(s)

Brenton Kenkel and Curtis S. Signorino

See Also

model.matrix.polywog for constructing the design matrix.

model.matrix.polywog *Model matrix of a polywog model*

Description

Constructs the design matrix used to fit a polywog model, similar to model.matrix.lm.

Usage

```
## S3 method for class 'polywog'
model.matrix(object, type = c("raw", "expanded"), ...)
```

Arguments

`object`	a fitted model of class `"polywog"`
`type`	`"raw"`, the default, returns the non-expanded model matrix with no intercept (same number of columns as `object$polyTerms`). `"expanded"` returns the polynomial expansion used in fitting (number of columns equals `length(object$coefficients)`).
`...`	other arguments to be passed to further methods (typically only used internally)

Details

There are two types of model matrix a user might want to construct. First, there is the matrix of the raw input terms that go into the eventual polynomial expansion. Such a matrix can be obtained by using `type = "raw"` (the default). The other form of the model matrix is the full polynomial expansion, where each column contains some power of the raw inputs. This can be obtained by using `type = "expanded"`.

Value

The design matrix of the specified model, consisting either of raw terms or the full polynomial expansion depending on the `type` argument.

Author(s)

Brenton Kenkel and Curtis S. Signorino

`plot.margEff.polywog` *Plot marginal effects*

Description

Generates density plots of the observationwise marginal effects computed by `margEff.polywog`.

Usage

```
## S3 method for class 'margEff.polywog'
plot(x, ...)
```

Arguments

`x`	output of `margEff.polywog`.
`...`	plotting parameters to be passed to `plot.density`.

Value

Data frame containing the variables whose densities were plotted, invisibly.

Author(s)

Brenton Kenkel and Curtis S. Signorino

plot.polywog *Univariate and bivariate fitted value plots*

Description

Generates plots of the relationship between input variables and the expected value of the outcome, using predVals as a backend.

Usage

```
## S3 method for class 'polywog'
plot(
  x,
  which = NULL,
  ask = FALSE,
  auto.set.par = TRUE,
  interval = TRUE,
  level = 0.95,
  FUN3D = c("contour", "filled.contour", "wireframe", "persp3d"),
  control.plot = list(),
  ...
)
```

Arguments

x	a fitted model of class "polywog", typically the output of polywog.
which	selection of variables to plot: a character vector containing one or two names of raw input variables (see x$varNames). May also be a numeric vector corresponding to indices of x$varNames. If which = NULL, a plot of each individual term will be generated.
ask	logical: whether to display an interactive menu of terms to select.
auto.set.par	logical: whether to temporarily change the graphics parameters so that multiple plots are displayed in one window (e.g., each univariate plot when which = NULL).
interval	logical: whether to display bootstrap confidence intervals around each fitted value. Not available for bivariate plots unless FUN3d = "persp3d".
level	confidence level for the intervals.
FUN3D	which plotting function to use to generate bivariate plots. Valid options include "contour" (the default) and "filled.contour"; "wireframe", which requires the **lattice** package; and "persp3d", which requires the **rgl** package.
control.plot	list of arguments to be passed to the underlying plotting functions (e.g., axis labels and limits).
...	additional arguments to be passed to predVals.

Details

By default, a univariate plot generated by `plot.polywog` shows the relationship between the selected input variable and the expected outcome while holding all other covariates at "central" values (as in `predVals`). The values that the other variables are held out can be changed by supplying additional arguments to ..., as in the examples below.

Similarly, a bivariate plot shows the relationship between two input variables and the expected outcome while holding all else fixed. If either variable is binary or categorical, the plot will show the relationship between one variable and the expected outcome across each value/level of the other.

Value

An object of class `preplot.polywog`, invisibly. This is a data frame generated by `predVals` that contains all information used in plotting.

Author(s)

Brenton Kenkel and Curtis S. Signorino

Examples

```
## Using occupational prestige data
data(Prestige, package = "carData")
Prestige <- transform(Prestige, income = income / 1000)

## Fit a polywog model with bootstrap iterations
## (note: using low convergence threshold to shorten computation time of the
## example, *not* recommended in practice!)
set.seed(22)
fit1 <- polywog(prestige ~ education + income + type,
                data = Prestige,
                degree = 2,
                boot = 5,
                thresh = 1e-4)

## All univariate relationships
plot(fit1, n = 20)

## Predicted prestige across occupational categories
plot(fit1, which = "type",
     control.plot = list(xlab = "occupational category"))

## Predicted prestige by education across occupational categories
plot(fit1, which = c("education", "type"), n = 20)

## Joint effect of education and income
plot(fit1, which = c("education", "income"), n = 10)

## Bring up interactive menu
## Not run:
plot(fit1, ask = TRUE)
```

```
  # displays menu:
  # Select one or two variable numbers (separated by spaces), or 0 to exit:

  # 1: education
  # 2: income
  # 3: type

## End(Not run)
```

---

polywog *Polynomial regression with oracle variable selection*

Description

Fits a regression model using a polynomial basis expansion of the input variables, with penalization via the adaptive LASSO or SCAD to provide oracle variable selection.

Usage

```
polywog(
  formula,
  data,
  subset,
  weights,
  na.action,
  degree = 3,
  family = c("gaussian", "binomial"),
  method = c("alasso", "scad"),
  penwt.method = c("lm", "glm"),
  unpenalized = character(0),
  .parallel = FALSE,
  boot = 0,
  control.boot = control.bp(.parallel = .parallel),
  lambda = NULL,
  nlambda = 100,
  lambda.min.ratio = 1e-04,
  nfolds = 10,
  foldid = NULL,
  thresh = ifelse(method == "alasso", 1e-07, 0.001),
  maxit = ifelse(method == "alasso", 1e+05, 5000),
  model = TRUE,
  X = FALSE,
  y = FALSE
)
```

Arguments

formula	model formula specifying the response and input variables. See "Details" for more information.
data	a data frame, list or environment containing the variables specified in the model formula.
subset	an optional vector specifying a subset of observations to be used in fitting.
weights	an optional vector specifying weights for each observation to be used in fitting.
na.action	a function specifying what to do with observations containing NAs (default na.omit).
degree	integer specifying the degree of the polynomial expansion of the input variables.
family	"gaussian" (default) or "binomial" for logistic regression (binary response only).
method	variable selection method: "alasso" (default) for adaptive LASSO or "scad" for SCAD. You can also select method = "none" to return the model matrix and other information without fitting.
penwt.method	estimator for obtaining first-stage estimates in logistic models when method = "alasso": "lm" (default) for a linear probability model, "glm" for logistic regression.
unpenalized	names of model terms to be exempt from the adaptive penalty (only available when method = "alasso").
.parallel	logical: whether to perform k-fold cross-validation in parallel (only available when method = "alasso"). See "Details" below for more information on parallel computation.
boot	number of bootstrap iterations (0 for no bootstrapping).
control.boot	list of arguments to be passed to bootPolywog when bootstrapping; see control.bp.
lambda	a vector of values from which the penalty factor is to be selected via k-fold cross-validation. lambda is left unspecified by default, in which case a sequence of values is generated automatically, controlled by the nlambda and lambda.min.ratio arguments. Naturally, k-fold cross-validation is skipped if lambda contains exactly one value.
nlambda	number of values of the penalty factor to examine via cross-validation if lambda is not specified in advance; see "Details".
lambda.min.ratio	ratio of the lowest value to the highest in the generated sequence of values of the penalty factor if lambda is not specified; see "Details".
nfolds	number of folds to use in cross-validation to select the penalization factor.
foldid	optional vector manually assigning fold numbers to each observation used for fitting (only available when method = "alasso").
thresh	convergence threshold, passed as the thresh argument to glmnet when method = "alasso" and as the eps argument to ncvreg when method = "scad".
maxit	maximum number of iterations to allow in adaptive LASSO or SCAD fitting.
model	logical: whether to include the model frame in the returned object.
X	logical: whether to include the raw design matrix (i.e., the matrix of input variables prior to taking their polynomial expansion) in the returned object.
y	logical: whether to include the response variable in the returned object.

Details

The design matrix for the regression is a polynomial basis expansion of the matrix of raw input variables. This includes all powers and interactions of the input variables up to the specified `degree`. For example, the following terms will be included in `polywog(y ~ x1 + x2, degree = 3, ...)`:

- terms of degree 0: intercept
- terms of degree 1: `x1`, `x2`
- terms of degree 2: `x1^2`, `x2^2`, `x1*x2`
- terms of degree 3: `x1^3`, `x2^3`, `x1*x2^2`, `x1^2*x2`

To exclude certain terms from the basis expansion, use a model formula like `y ~ x1 + x2 | z1 + z2`. Only the degree 1 terms of `z1` and `z2` will be included.

It is possible that the "raw" basis expansion will be rank-deficient, such as if there are binary input variables (in which case $x_i = x_i^n$ for all $n > 0$). The procedure detects collinearity via `qr` and removes extraneous columns before fitting.

For both the adaptive LASSO and SCAD, the penalization factor λ is chosen by k-fold cross-validation. The selected value minimizes the average mean squared error of out-of-sample fits. (To select both λ and the polynomial degree simultaneously via cross-validation, see `cv.polywog`.)

The cross-validation process may be run in parallel via `foreach` by registering an appropriate backend and specifying `.parallel = TRUE`. The appropriate backend is system-specific; see `foreach` for information on selecting and registering a backend. The bootstrap iterations may also be run in parallel by specifying `control.boot = control.bp(.parallel = TRUE)`.

Value

An object of class `"polywog"`, a list containing:

`coefficients` the estimated coefficients.

`lambda` value of the penalty factor λ used to fit the final model.

`lambda.cv` a list containing the results of the cross-validation procedure used to select the penalty factor:

- `lambda` values of the penalty factor tested in cross-validation.
- `cvError` out-of-fold prediction error corresponding to each value of `lambda`.
- `lambdaMin` value of `lambda` with the minimal cross-validation error.
- `errorMin` minimized value of the cross-validation error.

`fitted.values` the fitted mean values for each observation used in fitting.

`lmcoef` coefficients from an unpenalized least-squares regression of the response variable on the polynomial expansion of the input variables.

`penwt` adaptive weight given to each term in the LASSO penalty (`NULL` for models fit via SCAD).

`formula` model formula, as a `Formula` object.

`degree` degree of the polynomial basis expansion.

`family` model family, `"gaussian"` or `"binomial"`.

`weights` observation weights if specified.

`method` the specified regularization method.

`penwt.method` the specified method for calculating the adaptive LASSO weights (`NULL` for models fit via SCAD).

`unpenalized` logical vector indicating which terms were not included in the LASSO penalty.

`thresh` convergence threshold used in fitting.

`maxit` iteration limit used in fitting.

`terms` the `terms` object used to construct the model frame.

`polyTerms` a matrix indicating the power of each raw input term (columns) in each term of the polynomial expansion used in fitting (rows).

`nobs` the number of observations used to fit the model.

`na.action` information on how `NA` values in the input data were handled.

`xlevels` levels of factor variables used in fitting.

`varNames` names of the raw input variables included in the model formula.

`call` the original function call.

`model` if `model = TRUE`, the model frame used in fitting; otherwise `NULL`.

`X` if `X = TRUE`, the raw model matrix (i.e., prior to taking the polynomial expansion); otherwise `NULL`. For calculating the expanded model matrix, see `model.matrix.polywog`.

`y` if `y = TRUE`, the response variable used in fitting; otherwise `NULL`.

`boot.matrix` if `boot > 0`, a sparse matrix of class `"dgCMatrix-class"` where each column is the estimate from a bootstrap replicate. See `bootPolywog` for more information on bootstrapping.

Author(s)

Brenton Kenkel and Curtis S. Signorino

References

Brenton Kenkel and Curtis S. Signorino. 2012. "A Method for Flexible Functional Form Estimation: Bootstrapped Basis Regression with Variable Selection." Typescript, University of Rochester.

See Also

To estimate variation via the bootstrap, see `bootPolywog`. To generate fitted values, see `predVals` (and the underlying method `predict.polywog`). For plots, see `plot.polywog`. The polynomial degree may be selected via cross-validation using `cv.polywog`.

Adaptive LASSO estimates are provided via `glmnet` and `cv.glmnet` from the **glmnet** package. SCAD estimates are via `ncvreg` and `cv.ncvreg` in the **ncvreg** package.

Examples

```
## Using occupational prestige data
data(Prestige, package = "carData")
Prestige <- transform(Prestige, income = income / 1000)

## Fit a polywog model with bootstrap iterations
```

```
## (note: using low convergence threshold to shorten computation time of the
## example, *not* recommended in practice!)
set.seed(22)
fit1 <- polywog(prestige ~ education + income + type,
                data = Prestige,
                degree = 2,
                boot = 5,
                thresh = 1e-4)

## Basic information
print(fit1)
summary(fit1)

## See how fitted values change with education holding all else fixed
predVals(fit1, "education", n = 10)

## Plot univariate relationships
plot(fit1)

## Use SCAD instead of adaptive LASSO
fit2 <- update(fit1, method = "scad", thresh = 1e-3)
cbind(coef(fit1), coef(fit2))
```

`predict.polywog` *Predict method for polywog objects*

Description

Generates fitted values, including bootstrap confidence intervals, for in- and out-of-sample data from a fitted polywog model.

Usage

```
## S3 method for class 'polywog'
predict(
  object,
  newdata,
  type = c("link", "response"),
  interval = FALSE,
  level = 0.95,
  bag = FALSE,
  na.action = na.pass,
  ...
)
```

Arguments

`object`	a fitted model of class `"polywog"`, typically the output of `polywog`.

newdata	an optional data frame containing observations for which fitted values should be computed. If not specified, fitted values are generated for the data used to fit the model.
type	specifies whether the fitted values should be generated on the link scale ($X\beta$) or in terms of the expected value of the response variable. These only differ for binomial family models.
interval	logical: whether to calculate bootstrap confidence intervals for each fitted value.
level	confidence level for the intervals.
bag	logical: whether to use "bootstrap aggregation" to generate the main fitted values (if FALSE, they are calculated from the main model fit).
na.action	a function specifying what to do with observations in newdata containing NAs (default na.pass). See "Details".
...	other arguments, currently ignored.

Value

If interval = TRUE, a matrix containing each fitted value and its confidence interval. Otherwise, a vector containing the fitted values.

Author(s)

Brenton Kenkel and Curtis S. Signorino

See Also

For more user-friendly generation of fitted values, see predVals. To compute marginal effects, see margEff.polywog.

---

predVals *Easy computation of fitted values*

Description

User-friendly generation of fitted values and their confidence intervals from models of class "polywog", using the "observed-value approach" advocated by Hanmer and Kalkan (2013).

Usage

```
predVals(
  model,
  xvars,
  data = model$model,
  xlims = list(),
  n = 100,
  interval = TRUE,
  level = 0.95,
```

```
  maxrows = 10000,
  report = FALSE,
  .parallel = FALSE,
  ...
)
```

Arguments

model	a fitted model of class "polywog", typically the output of polywog.
xvars	a character vector containing names of raw input variables (from model$varNames). Partial matches are allowed.
data	data frame to treat as the observed sample (defaults to the data used to fit the supplied model)
xlims	named list of limits for the evaluation grid for each continuous variable in xvars. If not given, the variable's observed range is used.
n	number of grid points at which to evaluate each continuous variable in xvars.
interval	logical: whether to compute bootstrap confidence intervals for each fitted value.
level	confidence level for the intervals.
maxrows	maximum number of rows of output. Used to prevent accidental memory overruns when xvars contains more than two continuous variables.
report	logical: whether to print a status bar. Not available if .parallel = TRUE.
.parallel	logical: whether to perform bootstrap iterations in parallel using foreach. See the "Details" section of the bootPolywog documentation page for more on parallel computation.
...	other arguments, currently ignored

Details

predVals allows users to examine the estimated effects of input variables on the expected outcome using the coefficients returned by polywog. The procedure is designed so that, for a preliminary analysis, the user can simply specify the fitted model and the independent variable of interest, and quickly obtain predicted values.

The predicted values are generated according to Hanmer and Kalkan's (2013) observed-value approach, which takes the form of a nested loop. When xvars contains a single variable X_m, the procedure is as follows:

1. For each level x of X_m in data (if X_m is discrete) or each element x of a grid over the range of X_m in data (if X_m is continuous):
   (a) For each observation i of data:
      i. Set $X_{mi} = x$, while holding all other variables X_{-mi} at their observed levels
      ii. Compute the predicted value of Y_i for the modified observation i, using the estimated model coefficients (as in predict.polywog)
   (b) The predicted value of Y given $X_m = x$ is the average of the predictions computed in the previous step

This observed-value approach provides a better estimate of population average effects for nonlinear models than does the traditional approach, which is to vary X_m across its levels/range while holding each other covariate to its mean or median in `data` (Hanmer and Kalkan 2013).

When `xvars` consists of multiple variables $X_1, \ldots, X_M$, the `predVals` procedure is the same, except the outer loop is over every *combination* of their levels in `data`.

All confidence intervals are generated via the bootstrap. Specifically, `predVals` repeats the above procedure for each set of bootstrap coefficients and computes order statistics of the resulting set of averages (for each combination of levels of `xvars`). If `model` does not have a `boot.matrix` element (see `bootPolywog`), confidence intervals will not be computed.

Value

A data frame containing the fitted values and confidence intervals (if requested) for each combination of covariate values.

The returned data frame also inherits from class `"preplot.polywog"`. This is used by `plot.polywog`, which calls `predVals` to compute the values to plot.

Author(s)

Brenton Kenkel and Curtis S. Signorino

References

Michael J. Hanmer and Kerem Ozan Kalkan. 2013. "Behind the Curve: Clarifying the Best Approach to Calculating Predicted Probabilities and Marginal Effects from Limited Dependent Variable Models." *American Journal of Political Science* 57(1):263–277.

See Also

`predict.polywog` for more flexible (but less user-friendly) computation of fitted values. `plot.polywog` for plotting fitted values and their confidence intervals.

Examples

```
## Using occupational prestige data
data(Prestige, package = "carData")
Prestige <- transform(Prestige, income = income / 1000)

## Fit a polywog model with bootstrap iterations
## (note: using low convergence threshold to shorten computation time of the
## example, *not* recommended in practice!)
set.seed(22)
fit1 <- polywog(prestige ~ education + income + type,
                data = Prestige,
                degree = 2,
                boot = 5,
                thresh = 1e-4)

## Predicted prestige across occupational categories
predVals(fit1, "type")
```

```
## Predicted prestige by education
predVals(fit1, "education", n = 10)

## Plotting
pred_income <- predVals(fit1, "income", n = 10)
plot(pred_income)
```

`summary.margEff.polywog` *Summarize marginal effects*

Description

Generates a table of the average marginal effects and quartiles (or other order statistics if requested) from a `"margEff.polywog"` object.

Usage

```
## S3 method for class 'margEff.polywog'
summary(object, probs = seq(0, 1, by = 0.25), ...)
```

Arguments

`object`	output of `margEff.polywog`.
`probs`	order statistics to display.
`...`	other arguments, currently ignored.

Value

Table of results.

Author(s)

Brenton Kenkel and Curtis S. Signorino

summary.polywog *Summarize a fitted polywog model*

Description

Generates a "regression table" to summarize the fitted model, including coefficients along with their bootstrapped standard errors and confidence intervals. If the fitted model does not have a `boot.matrix` element, the output will contain `NA`s for the standard errors, and confidence intervals will not be displayed.

Usage

```
## S3 method for class 'polywog'
summary(object, level = 0.95, prop0 = FALSE, ...)
```

Arguments

`object`	a fitted model of class `"polywog"`, typically the output of `polywog`.
`level`	width of the bootstrap confidence interval to compute for the model coefficients.
`prop0`	logical: whether to print the proportion of bootstrap iterations in which each coefficient was estimated as exactly 0. This may be informative but should *not* be interpreted as a p-value.
`...`	other arguments, currently ignored.

Value

An object of class `"summary.polywog"` whose elements are the "regression table" (`coefficients`) and additional information from the original fitted model.

Author(s)

Brenton Kenkel and Curtis S. Signorino

Index