0.1 probit.gam: Generalized Additive Model for Dichotomous Dependent Variables

This function runs a nonparametric Generalized Additive Model (GAM) for dichotomous dependent variables.

Syntax

```
> z.out <- zelig(y ~ x1 + s(x2), model = "probit.gam", data = mydata)
> x.out <- setx(z.out)
> s.out <- sim(z.out, x = x.out)
```

Where `s()` indicates a variable to be estimated via nonparametric smooth. All variables for which `s()` is not specified, are estimated via standard parametric methods.

Additional Inputs

In addition to the standard inputs, `zelig()` takes the following additional options for GAM models.

- `method`: Controls the fitting method to be used. Fitting methods are selected via a list environment within `method=gam.method()`. See `gam.method()` for details.
- `scale`: Generalized Cross Validation (GCV) is used if `scale = 0` (see the "Model" section for details) except for Logit models where a Un-Biased Risk Estimator (UBRE) (also see the "Model" section for details) is used with a scale parameter assumed to be 1. If `scale` is greater than 1, it is assumed to be the scale parameter/variance and UBRE is used. If `scale` is negative GCV is used.
- `knots`: An optional list of knot values to be used for the construction of basis functions.
- `H`: A user supplied fixed quadratic penalty on the parameters of the GAM can be supplied with this as its coefficient matrix. For example, ridge penalties can be added to the parameters of the GAM to aid in identification on the scale of the linear predictor.
- `sp`: A vector of smoothing parameters for each term.
- `...`: additional options passed to the `probit.gam` model. See the `mgcv` library for details.

Examples

1. Basic Example

   Create some count data:

```
> set.seed(0);  n <- 400; sig <- 2;
> x0 <- runif(n, 0, 1);  x1 <- runif(n, 0, 1)
> x2 <- runif(n, 0, 1);  x3 <- runif(n, 0, 1)
> g <- (f-5)/3
> g <- binomial()$linkinv(g)
> y <- rbinom(g,1,g)
> my.data <- as.data.frame(cbind(y, x0, x1, x2, x3))
```

Estimate the model, summarize the results, and plot nonlinearities:

```
> z.out <- zelig(y ~ s(x0) + s(x1) + s(x2) + s(x3), model = "probit.gam",
+     data = my.data)
> summary(z.out)
> plot(z.out, pages = 1, residuals = TRUE)
```

Note that the `plot()` function can be used after model estimation and before simulation to view the nonlinear relationships in the independent variables:

Set values for the explanatory variables to their default (mean/mode) values, then simulate, summarize and plot quantities of interest:

```
> x.out <- setx(z.out)
> s.out <- sim(z.out, x = x.out)
> summary(s.out)
> plot(s.out)
```

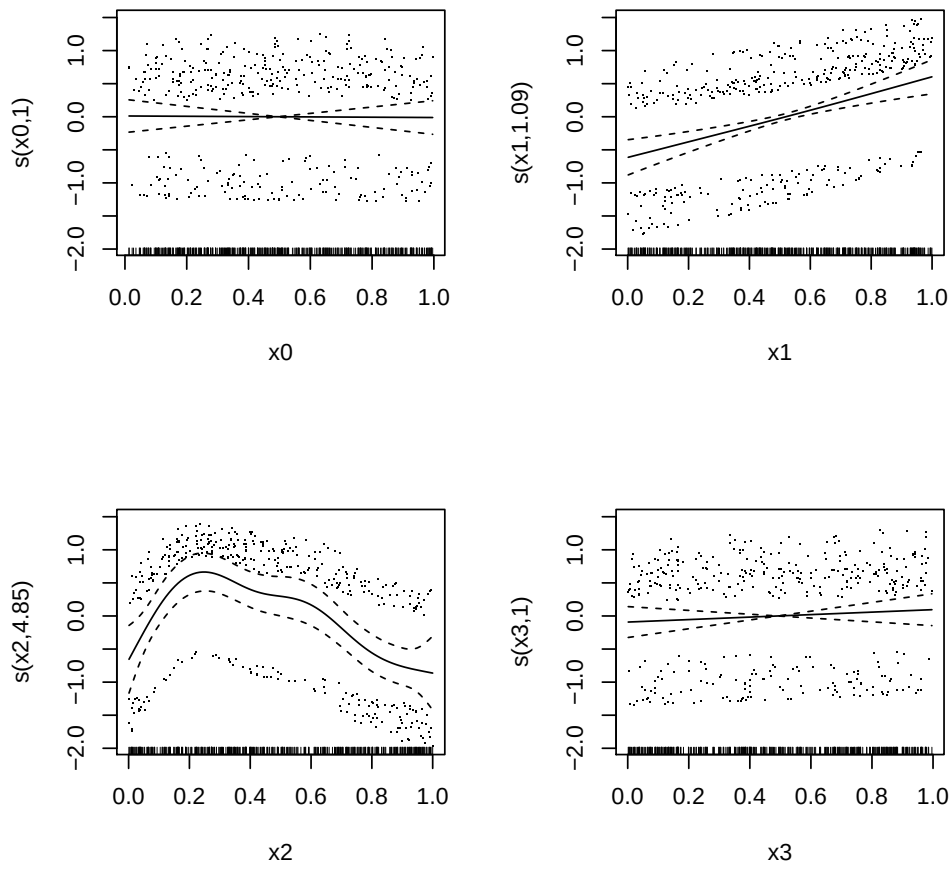
2. Simulating First Differences

   Estimating the risk difference (and risk ratio) between low values (20th percentile) and high values (80th percentile) of the explanatory variable `x3` while all the other variables are held at their default (mean/mode) values.

```
> x.high <- setx(z.out, x3 = quantile(my.data$x3, 0.8))
> x.low <- setx(z.out, x3 = quantile(my.data$x3, 0.2))
> s.out <- sim(z.out, x = x.high, x1 = x.low)
> summary(s.out)
> plot(s.out)
```

3. Variations in GAM model specification. Note that `setx` and `sim` work as shown in the above examples for any GAM model. As such, in the interest of parsimony, I will not re-specify the simulations of quantities of interest.

   An extra ridge penalty (useful with convergence problems):



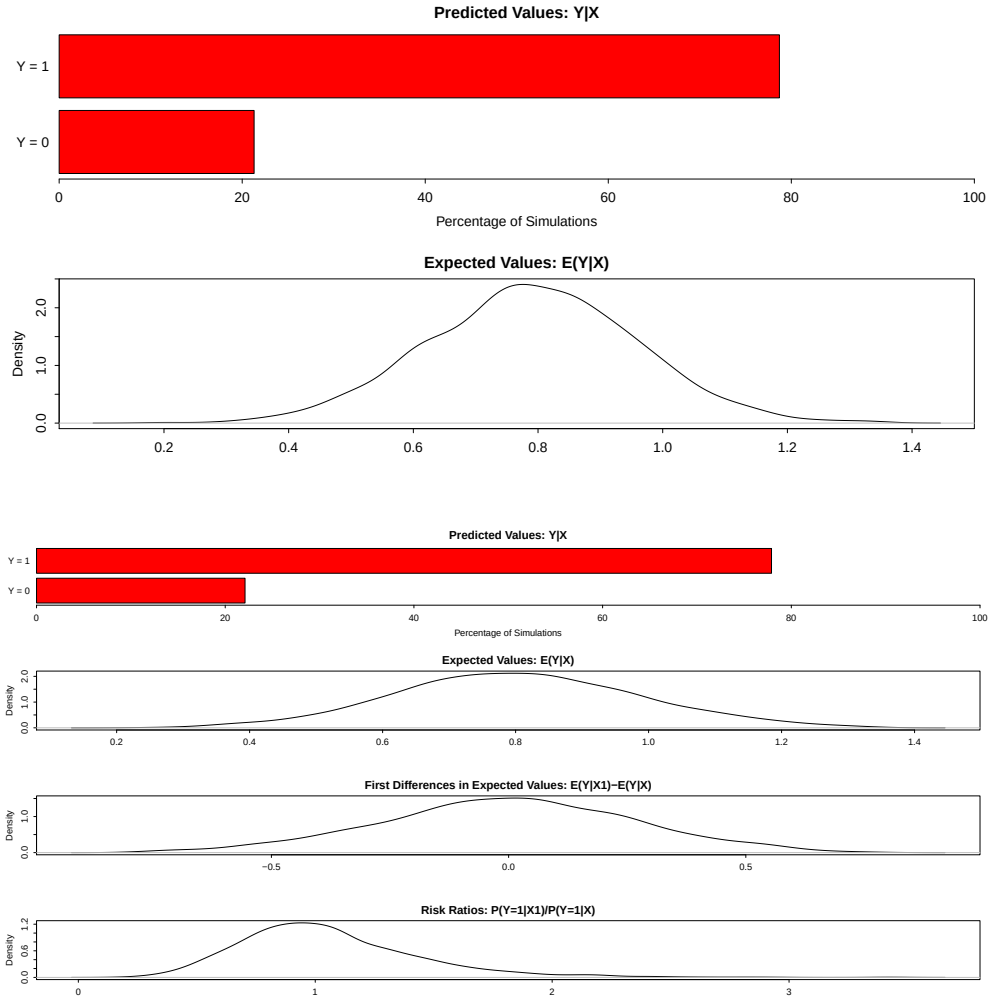

```
> z.out <- zelig(y ~ s(x0) + s(x1) + s(x2) + s(x3), H = diag(0.5,
+     37), model = "probit.gam", data = my.data)
> summary(z.out)
> plot(z.out, pages = 1, residuals = TRUE)
```

Set the smoothing parameter for the first term, estimate the rest:

```
> z.out <- zelig(y ~ s(x0) + s(x1) + s(x2) + s(x3), sp = c(0.01,
+     -1, -1, -1), model = "probit.gam", data = my.data)
> summary(z.out)
> plot(z.out, pages = 1)
```

Set lower bounds on smoothing parameters:

```
> z.out <- zelig(y ~ s(x0) + s(x1) + s(x2) + s(x3), min.sp = c(0.001,
+     0.01, 0, 10), model = "probit.gam", data = my.data)
```




```
> summary(z.out)
> plot(z.out, pages = 1)
```

A GAM with 3df regression spline term & 2 penalized terms:

```
> z.out <- zelig(y ~ s(x0, k = 4, fx = TRUE, bs = "tp") + s(x1,
+     k = 12) + s(x2, k = 15), model = "probit.gam", data = my.data)
> summary(z.out)
> plot(z.out, pages = 1)
```

Model

GAM models use families the same way GLM models do: they specify the distribution and link function to use in model fitting. In the case of `probit.gam` a normal link function is used. Specifically, let Y_i be the binary dependent variable for observation i which takes the value of either 0 or 1.

- The normal distribution has *stochastic component*

$$Y_i \sim \text{Bernoulli}(\pi_i)$$

  where $\pi_i = \Pr(Y_i = 1)$.

- The *systematic component* is given by:

$$\pi_i = \mathbf{\Phi}\left(x_i\beta + \sum_{j=1}^{J} f_j(Z_j)\right),$$

  where $\mathbf{\Phi}(\mu)$ is the cumulative distribution function of the Normal distribution with mean 0 and unit variance and $f_j(Z_j)$ for $j = 1, \ldots J$ is the set of smooth terms.

Generalized additive models (GAMs) are similar in many respects to generalized linear models (GLMs). Specifically, GAMs are generally fit by penalized maximum likelihood estimation and GAMs have (or can have) a parametric component identical to that of a GLM. The difference is that GAMs also include in their linear predictors a specified sum of smooth functions.

In this GAM implementation, smooth functions are represented using penalized regression splines. Two techniques may be used to estimate smoothing parameters: Generalized Cross Validation (GCV),

$$n\frac{D}{(n - DF)^2}, \tag{1}$$

or an Un-Biased Risk Estimator (UBRE) (which is effectively just a rescaled AIC),

$$\frac{D}{n} + 2s\frac{DF}{n - s}, \tag{2}$$

where D is the deviance, n is the number of observations, s is the scale parameter, and DF is the effective degrees of freedom of the model. The use of GCV or UBRE can be set by the user with the `scale` command described in the "Additional Inputs" section and in either case, smoothing parameters are chosen to minimize the GCV or UBRE score for the model.

Estimation for GAM models proceeds as follows: first, basis functions and a set (one or more) of quadratic penalty coefficient matrices are constructed for each smooth term. Second, a model matrix is is obtained for the parametric component of the GAM. These matrices are combined to produce a complete model matrix and a set of penalty matrices for the smooth terms. Iteratively Reweighted Least Squares (IRLS) is then used to estimate the model; at each iteration of the IRLS, a penalized weighted least squares model is run and the smoothing parameters of that model are estimated by GCV or UBRE. This process is repeated until convergence is achieved.

Further details of the GAM fitting process are given in Wood (2000, 2004, 2006).

Quantities of Interest

The quantities of interest for the `probit.gam` model are the same as those for the standard normal regression.

- The expected value (`qi$ev`) for the `probit.gam` model is the mean of simulations from the stochastic component,

$$\pi_i = \mathbf{\Phi}\left(x_i\beta + \sum_{j=1}^{J} f_j(Z_j)\right).$$

- The predicted values (`qi$pr`) are draws from the Binomial distribution with mean equal to the simulated expected value π_i.

- The first difference (`qi$fd`) for the probit.gam model is defined as

$$FD = \Pr(Y|w_1) - \Pr(Y|w)$$

  for $w = \{X, Z\}$.

Output Values

The output of each Zelig command contains useful information which you may view. For example, if you run `z.out <- zelig(y ~ x, model = "probit.gam", data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using `coefficients(z.out)`, and a default summary of information through `summary(z.out)`. Other elements available through the `$` operator are listed below.

- From the `zelig()` output stored in `z.out`, you may extract:
  - `coefficients`: parameter estimates for the explanatory variables.
  - `fitted.values`: the vector of fitted values for the explanatory variables.
  - `residuals`: the working residuals in the final iteration of the IRLS fit.
  - `linear.predictors`: the vector of $x_i\beta$.
  - `aic`: Akaike's Information Criterion (minus twice the maximized log-likelihood plus twice the number of coefficients).
  - `method`: the fitting method used.
  - `converged`: logical indicating weather the model converged or not.
  - `smooth`: information about the smoothed parameters.
  - `df.residual`: the residual degrees of freedom.
  - `df.null`: the residual degrees of freedom for the null model.
  - `data`: the input data frame.

  - `model`: the model matrix used.

- From `summary(z.out)`(as well as from `zelig()`), you may extract:
  - `p.coeff`: the coefficients of the parametric components of the model.
  - `se`: the standard errors of the entire model.
  - `p.table`: the coefficients, standard errors, and associated t statistics for the parametric portion of the model.
  - `s.table`: the table of estimated degrees of freedom, estimated rank, F statistics, and p-values for the nonparametric portion of the model.
  - `cov.scaled`: a $k \times k$ matrix of scaled covariances.
  - `cov.unscaled`: a $k \times k$ matrix of unscaled covariances.

- From the `sim()` output stored in `s.out`, you may extract:
  - `qi$ev`: the simulated expected probabilities for the specified values of `x`.
  - `qi$pr`: the simulated predicted values for the specified values of `x`.
  - `qi$fd`: the simulated first differences in the expected probabilities simulated from `x` and `x1`.

How to Cite

To cite the *probit.gam* Zelig model:

> Skyler J. Cranmer. 2007. "probit.gam: Generalized Additive Model for Dichotomous Dependent Variables" in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"`http://gking.harvard.edu/zelig`

To cite Zelig as a whole, please reference these two sources:

> Kosuke Imai, Gary King, and Olivia Lau. 2007. "Zelig: Everyone's Statistical Software," `http://GKing.harvard.edu/zelig`.

> Imai, Kosuke, Gary King, and Olivia Lau. (2008). "Toward A Common Framework for Statistical Analysis and Development." Journal of Computational and Graphical Statistics, Vol. 17, No. 4 (December), pp. 892-913.

See also

The `gam.logit` model is adapted from the `mgcv` package by Simon N. Wood (**?**). Advanced users may wish to refer to `help(gam)`, **?**, **?**, and other documentation accompanying the `mgcv` package. All examples are reproduced and extended from `mgcv`'s `gam()` help pages.