Package ‘wsbackfit’

July 22, 2025

Type Package

Title Weighted Smooth Backfitting for Structured Models

Version 1.0-5

Date 2021-04-30

Imports graphics, stats

Description Non- and semiparametric regression for generalized additive, partial linear, and varying coefficient models as well as their combinations via smoothed backfitting. Based on Roca-Pardinas J and Sperlich S (2010) <doi:10.1007/s11222-009-9130-2>; Mammen E, Linton O and Nielsen J (1999) <doi:10.1214/aos/1017939138>; Lee YK, Mammen E, Park BU (2012) <doi:10.1214/12-AOS1026>.

License GPL

LazyLoad yes

Depends R (>= 3.5.0)

NeedsCompilation yes

Author Javier Roca-Pardinas [aut, cre],
Maria Xose Rodriguez-Alvarez [aut],
Stefan Sperlich [aut],
Alan Miller (FORTRAN code lsq.f90: weighted least-squares module) [ctb]

Maintainer Javier Roca-Pardinas <`roca@uvigo.es`>

Repository CRAN

Date/Publication 2021-05-04 16:50:02 UTC

Contents

`wsbackfit-package` *Weighted Smooth Backfitting for Structured Models*

Description

Non- and semiparametric regression for generalized additive, partial linear, and varying coefficient models as well as their combinations. Specifically, the package provides estimation procedures for a large class of regression models common in applied statistics. The regression models belong to the class of the so-called generalized structured models, i.e.,

$$E[Y|X,Z] = G(g_0 + \sum_j g_j(X_j)Z_j + Z_k^{'}\beta).$$

Note that, up to identification restrictions specified e.g. in Park and Mammen (2006), several of the X_j and Z_j can refer to the same variable. For example, all X_j may be the same but all Z_j different.

The estimation procedure is based on smoothed backfitting which to our knowledge is the statistically most efficient existing procedure for this model class. Additional weights allow sampling weights, trimming, or efficient estimation under heteroscedasticity. This package also allows to either set the bandwidths or automatically select them using k-fold cross-validation. The option 'offset' facilitates the application of smooth backfitting on aggregated data.

Details

Package:	wsbackfit
Type:	Package
Version:	1.0-5
Date:	2021-04-30
License:	GPL

Author(s)

Javier Roca-Pardinas, Maria Xose Rodriguez-Alvarez, Stefan Sperlich

Maintainer: Javier Roca-Pardinas <roca@uvigo.es>

References

Han, K. and Park B.U. (2018). Smooth backfitting for errors-in-variables additive models. Annals of Statistics, 46, 2216-2250.

Lee, Y.K., Mammen, E. and Park, B.U. (2012). Flexible generalized varying coefficient regression models. The Annals of Statistics, 40(3), 1906-1933.

Mammen, E. and Nielsen, J. (2003). Generalised structured models. Biometrika, 90, 551-566.

Mammen, E. Linton, O. and Nielsen, J. (1999). The existence and asymptotic properties of a backfitting projection algorithm under weak conditions. Annals of Statistics, 27 (5), 1443-1490.

Mammen, E. and Park, B.U. (2006). A simple smooth backfitting method for additive models. Annals of Statistics, 34 (5), 2252-2271.

Nielsen. J. and Sperlich, S. (2005). Smooth backfitting in practice. Journal of the Royal Statistical Society, B, 67, 43-61.

Roca-Pardinas, J. and Sperlich, S. (2010). Feasible Estimation in Generalized Structured Models. Statistics and Computing, 20, 367-379.

`infect` *Postoperative Infection Data.*

Description

Data from a prospective study conducted at the University Hospital of Santiago de Compostela (Spain). A total of 2318 patients who underwent surgery at this center between January 1996 and March 1997 were characterized post-operatively, in respect of whether they suffered or not post-operative infection.

Usage

```
data(infect)
```

Format

A data frame with 2318 observations on the following 6 variables.

- `age` patient's age.
- `sex` patient's sex. Coded as 1 = `Man` and 2 = `Woman`.
- `linf` lymphocytes (expressed as relative counts (in %) of the white blood cell count)
- `gluc` plasma glucose concentration (measured in mg/dl)
- `diab` diabetes. Coded as 1 = presence and 2 = absence.
- `inf` variable indicating whether the patient suffered (inf = 1) or not (inf = 0) a post-operative infection.

Examples

```
data(infect)
summary(infect)
```

`plot.sback` *Default sback plotting*

Description

Takes a fitted object produced by `sback()` and plots the estimates of the nonparametric functions on the scale of their respective covariates, no matter whether a particular nonparametric function is an additive component or a varying coefficient.

Usage

```
## S3 method for class 'sback'
plot(x, composed = TRUE, ask = TRUE, select = NULL, ...)
```

Arguments

`x`	an object of class `sback` as produced by `sback()`.
`composed`	a logical value. If `TRUE`, the default, the function plots the estimates of the composed (linear plus nonlinear) nonparametric functions (see Details).
`ask`	a logical value. If `TRUE`, the default, the user is asked for confirmation, before a new figure is drawn.
`select`	Allows the plot for a single model term to be selected for printing. e.g. if you just want the plot for the second smooth term set select = 2.
`...`	other graphics parameters to pass on to plotting commands.

Details

For identifiability purposes, the estimating algorithm implemented in the `wsbackfit` package decomposes each nonparametric function in two components: a linear (parametric) component and a nonlinear (nonparametric) component. For plotting, the user can choose to plot these components either separately in one graph (`composed = FALSE`), or to only plot the resulting composed function (`composed = TRUE`). Also, for the varying coefficient terms, the plots show the estimated surface spanned by (g_j, X_j, Z_j).

Value

None

Author(s)

Javier Roca-Pardinas, Maria Xose Rodriguez-Alvarez and Stefan Sperlich

See Also

`sback`, `summary.sback`

Examples

```
library(wsbackfit)
################################################
# Gaussian Simulated Sample
###############################################
set.seed(123)
# Define the data generating process
n <- 1000
x1 <- runif(n)*4-2
x2 <- runif(n)*4-2
x3 <- runif(n)*4-2
x4 <- runif(n)*4-2
x5 <- as.numeric(runif(n)>0.6)

f1 <- 2*sin(2*x1)
f2 <- x2^2
f3 <- 0
f4 <- x4
f5 <- 1.5*x5

mu <- f1 + f2 + f3 + f4 + f5
err <- (0.5 + 0.5*x5)*rnorm(n)
y <- mu + err

df <- data.frame(x1 = x1, x2 = x2, x3 = x3, x4 = x4, x5 = as.factor(x5), y = y)

# Fit the model with a fixed bandwidth for each covariate
m0 <- sback(formula = y ~ x5 + sb(x1, h = 0.1) + sb(x2, h = 0.13)
  + sb(x3, h = 0.1) + sb(x4, h = 0.1), kbin = 30, data = df)

plot(m0)
```

`predict.sback` *Predict method for sback fits*

Description

Predicted smooth functions and values based on an sback object

Usage

```
## S3 method for class 'sback'
predict(object, newdata, newoffset = NULL, ...)
```

Arguments

`object`	an object of class `sback`.

`newdata`	a data frame containing the values of the covariates at which predictions are wanted. If not provided then the predictions correspond to the original data.
`newoffset`	an optional numerical vector containing an a priori known component to be included in the linear predictor for the predictions (offset associated with the newdata).
`...`	not yet implemented.

Value

A list with the following components:

`pdata`	the original supplied `newdata` argument.
`poffset`	the original supplied `newoffset` argument.
`coeff`	a numeric vector with the estimated coefficients. This vector includes both the parametric effects as well as the coefficients associated with the linear component of the nonparametric functions.
`peffects`	matrix with the estimated nonparametric functions (only the nonlinear component) for each covariate value in the original supplied newdata.
`pfitted.values`	a numeric vector with the fitted values for the supplied newdata.

Author(s)

Javier Roca-Pardinas, Maria Xose Rodriguez-Alvarez and Stefan Sperlich

See Also

`sback`, `summary.sback`

Examples

```
library(wsbackfit)
data(infect)

# Generalized varying coefficient model with binary response
m3 <- sback(formula = inf ~ sb(gluc, h = 10) + sb(gluc, by = linf, h = 10),
  data = infect, family = "binomial", kbin = 15)

summary(m3)

# Plot both linear and non linear
# components of nonparametric functions: composed = FALSE
op <- par(no.readonly = TRUE)
par(mfrow = c(1,3))
plot(m3, composed = FALSE)

# Personalized plots
# First obtain predictions in new data
# Create newdata for prediction
ngrid <- 30
```

```
gluc0 <- seq(50, 190, length = ngrid)
linf0 <- seq(0, 45, length = ngrid)
df  <- expand.grid(gluc = gluc0, linf = linf0)

m3p <- predict(m3, newdata = df)

par(mfrow = c(1,2))
ii <- order(df[,"gluc"])

## Parametric coefficients
names(m3p$coeff)

# Nonlinear components
colnames(m3p$peffects)

# Include the linear component
plot(df[ii,"gluc"], m3p$coeff[["gluc"]]*df[ii,"gluc"] +
  m3p$peffects[ii,"sb(gluc, h = 10)"],
  type = 'l', xlab = "Glucose (mg/dl)", ylab = "f_1(gluc)",
  main = "Nonparametric effect of Glucose")

# Include the linear component
plot(df[ii,"gluc"], m3p$coeff[["gluc:linf"]]*df[ii,"gluc"] +
  m3p$peffects[ii,"sb(gluc, h = 10, by = linf)"],
  type= 'l', xlab = "Glucose (mg/dl)", ylab = "f_2(gluc)",
  main = "Varying coefficients as a function of Glucose")

# Countour plot of the probability of post-opererational infection
n <- sqrt(nrow(df))
Z <- matrix(m3p$pfitted.values, n, n)
filled.contour(z = Z, x = gluc0, y = linf0,
  xlab = "Glucose (mg/dl)", ylab = "Lymphocytes (%)",
  main = "Probability of post-opererational infection")

par(op)
```

`print.sback` *Print a* sback *object.*

Description

The default print method for a sback object.

Usage

```
## S3 method for class 'sback'
print(x, ...)
```

Arguments

`x`	an object of class `sback` as produced by `sback`.
`...`	further arguments passed to or from other methods. Not yet implemented.

Value

None

Author(s)

Javier Roca-Pardinas, Maria Xose Rodriguez-Alvarez and Stefan Sperlich

See Also

`sback`, `summary.sback`, `plot.sback`

Examples

```
library(wsbackfit)
################################################
# Gaussian Simulated Sample
################################################
set.seed(123)
# Define the data generating process
n <- 1000
x1 <- runif(n)*4-2
x2 <- runif(n)*4-2
x3 <- runif(n)*4-2
x4 <- runif(n)*4-2
x5 <- as.numeric(runif(n)>0.6)

f1 <- 2*sin(2*x1)
f2 <- x2^2
f3 <- 0
f4 <- x4
f5 <- 1.5*x5

mu <- f1 + f2 + f3 + f4 + f5
err <- (0.5 + 0.5*x5)*rnorm(n)
y <- mu + err

df <- data.frame(x1 = x1, x2 = x2, x3 = x3, x4 = x4, x5 = as.factor(x5), y = y)

# Fit the model with a fixed bandwidth for each covariate
m0 <- sback(formula = y ~ x5 + sb(x1, h = 0.1) + sb(x2, h = 0.1)
  + sb(x3, h = 0.1) + sb(x4, h = 0.1), kbin = 30, data = df)

m0
```

`residuals.sback` *sback residuals*

Description

Returns residuals for a fitted sback object. Deviance, pearson, working and response residuals are available.

Usage

```
## S3 method for class 'sback'
residuals(object, type = c("deviance", "pearson", "working", "response"), ...)
```

Arguments

`object`	an object of class sback as produced by `sback`.
`type`	the type of residuals which should be returned: "deviance" (default), "pearson", "working" and "response".
`...`	further arguments passed to or from other methods. Not yet implemented.

Details

For details see `residuals.glm`.

Value

Numeric vector with the residuals.

Author(s)

Javier Roca-Pardinas, Maria Xose Rodriguez-Alvarez and Stefan Sperlich

See Also

`sback`, `summary.sback`, `plot.sback`.

Examples

```
library(wsbackfit)
data(infect)

# Generalized varying coefficient model with binary response
m3 <- sback(formula = inf ~ sb(gluc, h = 10) + sb(gluc, by = linf, h = 10),
  data = infect, family = "binomial", kbin = 15)

summary(m3)

# Deviance
```

```
summary(residuals(m3))

# Pearson
summary(residuals(m3, type = "pearson"))
```

---

sb — *Specify a nonparametric and/or a varying coefficient term in a wsbackfit formula*

Description

Function used to indicate nonparametric terms and varying coefficient terms in a `sback` formula.

Usage

```
sb(x1 = NULL, by = NULL, h = -1)
```

Arguments

`x1`	the univariate predictor
`by`	numeric predictor of the same dimension as `x1`. If present, the coefficients of this predictor depend, nonparametrically, on `x1`, i.e., a varying coefficient term.
`h`	bandwidth (on the scale of the predictor) for this term. If `h = -1`, the bandwidth is automatically selected using k-fold cross-validation (see `sback`). A value of 0 would indicate a linear fit. By default -1.

Value

A list with the following components:

`cov`	character vector with the name(s) of the involved predictor(s).
`h`	numeric value with the specified smoothing parameter.

Author(s)

Javier Roca-Pardinas, Maria Xose Rodriguez-Alvarez and Stefan Sperlich

See Also

`sback`, `summary.sback`, `plot.sback`

Examples

```
library(wsbackfit)
set.seed(123)
################################################
# Gaussian Simulated Sample
################################################
set.seed(123)
# Define the data generating process
n <- 1000
x1 <- runif(n)*4-2
x2 <- runif(n)*4-2
x3 <- runif(n)*4-2
x4 <- runif(n)*4-2
x5 <- as.numeric(runif(n)>0.6)

f1 <- 2*sin(2*x1)
f2 <- x2^2
f3 <- 0
f4 <- x4
f5 <- 1.5*x5

mu <- f1 + f2 + f3 + f4 + f5
err <- (0.5 + 0.5*x5)*rnorm(n)
y <- mu + err

df <- data.frame(x1 = x1, x2 = x2, x3 = x3, x4 = x4, x5 = as.factor(x5), y = y)

# Fit the model with a fixed bandwidth for each covariate
m0 <- sback(formula = y ~ x5 + sb(x1, h = 0.1) + sb(x2, h = 0.1)
  + sb(x3, h = 0.1) + sb(x4, h = 0.1), kbin = 30, data = df)

summary(m0)

op <- par(no.readonly = TRUE)

par(mfrow = c(2,2))
plot(m0)


# Fit the model with the bandwidths selected by k-fold cross-validation.
m1 <- sback(formula = y ~ x5 + sb(x1, h = -1) + sb(x2, h = -1)
  + sb(x3, h = -1) + sb(x4, h = -1), kbin = 30, bw.grid = seq(0.01, 0.99, length = 30),
  data = df)

summary(m1)

par(mfrow = c(2,2))
plot(m1)


par(op)
```

`sback` *Generalized additive and partially linear models*

Description

Main function for fitting generalized structured models by using smooth backfitting.

Usage

```
sback(formula, data, offset = NULL, weights = NULL,
  kernel = c("Gaussian", "Epanechnikov"),
  bw.grid = seq(0.01, 0.99, length = 30), c.bw.factor = FALSE,
  KfoldCV = 5, kbin = 30,
  family = c("gaussian", "binomial", "poisson"))
```

Arguments

`formula`	a `formula` object specifying the model to be fitted (see Details).
`data`	data frame representing the data and containing all needed variables
`offset`	an optional numerical vector containing priori known components to be included in the linear predictor during fitting. Default is zero.
`weights`	an optional numeric vector of 'prior weights' to be used in the fitting process. By default, the weights are set to one.
`kernel`	a character specifying the kernel function. Implemented are: Gaussian and Epanechnikov. By default 'Gaussian'.
`bw.grid`	numeric vector; a grid for for searching the bandwidth factor h_c when using cross-validation. The bandwidth for dimension (covariate) j is $h_c \sigma_j$, with σ_j being the standard deviation of X_j (see Details). Default is a sequence of length 30 between 0.01 and 0.99.
`c.bw.factor`	logical; indicates whether the common factor scheme for bandwidth selection proposed by Han and Park (2018) is performed. If TRUE, and provided the user has specified the (marginal) bandwidths for all nonparametric functions, say h_j, the functions searches for the common factor c_h that minimizes the deviance via (k-fold) cross-validation when the bandwidth used for dimension (covariate) j is $c_h h_j$. The search is done in an equispaced grid of length 15 between 0.5 and 1.5. The default is FALSE.
`KfoldCV`	number of cross-validation folds to be used for either (1) automatically selecting the optimal bandwidth (in the sequence given in argument `bw.grid`) for each nonparametric function; or (2) automatically selecting the optimal common bandwidth factor (see argument `c.bw.factor`). Default is 5.
`kbin`	an integer value specifying the number of binning knots. Default is 30.
`family`	a character specifying the distribution family. Implemented are: Gaussian, Binomial and Poisson. In all cases, the link function is the canonical one (logit for binomial, identity for Gaussian and logarithm for Poisson). By default 'gaussian'.

Details

The argument `formula` corresponds to the model for the conditional mean function, i.e.,

$$E[Y|X,Z] = G(g_0 + \sum_j g_j(X_j)Z_j + Z_k^{'}\beta).$$

This formula is similar to that used for the `glm` function, except that nonparametric functions can be added to the additive predictor by means of function `sb`. For instance, specification `y ~ x1 + sb(x2, h = -1)` assumes a parametric effect of `x1` (with `x1` either numerical or categorical), and a nonparametric effect of `x2`. `h = -1` indicates that the bandwidth should be selected using k-fold cross-validation. Varying coefficient terms get incorporated similarly. For example, `y ~ sb(x1, by = x2)` indicates that the coefficients of `x2` depend, nonparametrically, on `x1`. In this case both, `x1` and `x2`, should be numerical predictors.

With respect to the bandwidths associated with each nonparametric function specified using function `sb`, the user has two options: a) to specify in the formula the desired bandwidth - on the scale of the predictor - through argument `h` of function `sb` (followed or not by the common bandwidth factor scheme proposed by Han and Park (2018); see argument `c.bw.factor`); or, b) to allow the bandwidths to be automatically and data adaptively selected via cross-validation. In the latter case, the estimation procedure tests each of the bandwidth factors supplied in argument `bw.grid`, and selects the one that minimizes the deviance via (k-fold) cross-validation. The number `k` of cross-validation folds is specified through argument `KfoldCV`, with 5 by default. We note that when using cross-validation, to ensure that the bandwidths associated with the nonparametric functions are on the scale of the predictors, the finally used bandwidth is $h_j = h\sigma_j$ with σ_j being the standard deviation of X_j. That is, before fitting the model, each bandwidth factor h provided in `bw.grid` is multiplied by the standard deviation of the corresponding predictor. Note that the user has also the possibility to specify the bandwidths for some nonparametric function (through argument `h`), while letting for the remaining nonparametric functions the procedure select the bandwidths by cross-validation. For these functions, argument `h` should be set to `-1`. In this case, the common bandwidth factor scheme proposed by Han and Park (2018) cannot be used as it requires that all bandwidths are specified.

Finally, it is worth noting that for identifiability purposes, the estimating algorithm implemented in the `wsbackfit` package decomposes each nonparametric function in two components: a linear (parametric) component and a nonlinear (nonparametric) component. Note that it implies that for a varying coefficient term `~ sb(x1, by = x2)`, the parametric part includes the linear component associated with `x1`, as well as the linear interaction between `x1` and `x2`.

Value

A list with the following components:

`call`	the matched call.
`formula`	the original supplied `formula` argument.
`data`	the original supplied `data` argument.
`offset`	the original supplied `offset` argument.
`weights`	the original supplied `weights` argument.
`kernel`	the original supplied `kernel` argument.
`kbin`	the original supplied `kbin` argument.

`family`	the original supplied `family` argument.
`effects`	matrix with the estimated nonparametric functions (only the nonlinear component) for each covariate value in the original supplied data.
`fitted.values`	a numeric vector with the fitted values for the supplied data.
`residuals`	a numeric vector with the deviance residuals for the supplied data.
`h`	a numeric vector of the same length as the number of nonparametric functions, with the bandwidths actually used in the estimation (scaled. See Details).
`coeff`	a numeric vector with the estimated coefficients. This vector includes both the parametric effects as well as the coefficients associated with the linear component of the nonparametric functions.
`err.CV`	matrix with the cross-validated error (deviance) associated with the sequence of tested (unscaled) bandwidths. Each line corresponds to a particular bandwidth (unscaled. See Details).

Author(s)

Javier Roca-Pardinas, Maria Xose Rodriguez-Alvarez and Stefan Sperlich

References

Han, K. and Park B.U. (2018). Smooth backfitting for errors-in-variables additive models. Annals of Statistics, 46, 2216-2250.

See Also

`sb`, `print.sback`, `summary.sback`, `plot.sback`

Examples

```
library(wsbackfit)
###############################################
# Gaussian Simulated Sample
###############################################
set.seed(123)
# Define the data generating process
n <- 1000
x1 <- runif(n)*4-2
x2 <- runif(n)*4-2
x3 <- runif(n)*4-2
x4 <- runif(n)*4-2
x5 <- as.numeric(runif(n)>0.6)

f1 <- 2*sin(2*x1)
f2 <- x2^2
f3 <- 0
f4 <- x4
f5 <- 1.5*x5

mu <- f1 + f2 + f3 + f4 + f5
```

```
err <- (0.5 + 0.5*x5)*rnorm(n)
y <- mu + err

df <- data.frame(x1 = x1, x2 = x2, x3 = x3, x4 = x4, x5 = as.factor(x5), y = y)

# Fit the model with a fixed bandwidth for each covariate
m0 <- sback(formula = y ~ x5 + sb(x1, h = 0.1) + sb(x2, h = 0.1)
  + sb(x3, h = 0.1) + sb(x4, h = 0.1), kbin = 30, data = df)

summary(m0)

op <- par(no.readonly = TRUE)

par(mfrow = c(2,2))
plot(m0)

# Fit the model with bandwidths selectec using K-fold cross-validation
## Not run:
m0cv <- sback(formula = y ~ x5 + sb(x1) + sb(x2)
  + sb(x3) + sb(x4), kbin = 30, bw.grid = seq(0.01, 0.99, length = 30), KfoldCV = 5,
  data = df)

summary(m0cv)

par(mfrow = c(2,2))
plot(m0cv)

## End(Not run)

# Estimate Variance as a function of x5 (which is binary)
resid <- y - m0$fitted.values
sig0 <- var(resid[x5 == 0])
sig1 <- var(resid[x5 == 1])
w <- x5/sig1 + (1-x5)/sig0
m1 <- sback(formula = y ~ x5 + sb(x1, h = 0.1) + sb(x2, h = 0.1)
  + sb(x3, h = 0.1) + sb(x4, h = 0.1), weights = w, kbin = 30, data = df)

summary(m1)

par(mfrow = c(2,2))
plot(m1)

################################################
# Poisson Simulated Data
################################################
set.seed(123)
# Define the data generating process
n <- 1000

x1 <- runif(n,-1,1)
x2 <- runif(n,-1,1)

eta <- 2 + 3*x1^2 + 5*x2^3
```

```
exposure <- round(runif(n, 50, 500))
y <- rpois(n, exposure*exp(eta))
df <- data.frame(y = y, x1 = x1, x2 = x2)

# Fit the model
m2 <- sback(formula = y ~ sb(x1, h = 0.1) + sb(x2, h = 0.1),
  data = df, offset = log(exposure),
  kbin = 30, family = "poisson")

summary(m2)

par(mfrow = c(1,2))
plot(m2)

# Dataframe and offset for prediction
n.p <- 100
newoffset <- rep(0, n.p)
df.pred <- data.frame(x1 = seq(-1, 1,l = n.p), x2 = seq(-1, 1,l = n.p))

m2p <- predict(m2, newdata = df.pred, newoffset = newoffset)

###############################################
# Postoperative Infection Data
###############################################
data(infect)

# Generalized varying coefficient model with binary response
m3 <- sback(formula = inf ~ sb(gluc, h = 10) + sb(gluc, by = linf, h = 10),
  data = infect, family = "binomial", kbin = 15)

summary(m3)

# Plot both linear and non linear
# components of nonparametric functions: composed = FALSE
par(mfrow = c(1,3))
plot(m3, composed = FALSE)

# Personalized plots
# First obtain predictions in new data
# Create newdata for prediction
ngrid <- 30
gluc0 <- seq(50, 190, length = ngrid)
linf0 <- seq(0, 45, length = ngrid)
df  <- expand.grid(gluc = gluc0, linf = linf0)

m3p <- predict(m3, newdata = df)

par(mfrow = c(1,2))
ii <- order(df[,"gluc"])

## Parametric coefficients
names(m3p$coeff)
```

```
# Nonlinear components
colnames(m3p$peffects)

# Include the linear component
plot(df[ii,"gluc"], m3p$coeff[["gluc"]]*df[ii,"gluc"] +
  m3p$peffects[ii,"sb(gluc, h = 10)"],
  type = 'l', xlab = "Glucose (mg/dl)", ylab = "f_1(gluc)",
  main = "Nonparametric effect of Glucose")

# Include the linear component
plot(df[ii,"gluc"], m3p$coeff[["gluc:linf"]]*df[ii,"gluc"] +
  m3p$peffects[ii,"sb(gluc, h = 10, by = linf)"],
  type= 'l', xlab = "Glucose (mg/dl)", ylab = "f_2(gluc)",
  main = "Varying coefficients as a function of Glucose")

# Countour plot of the probability of post-opererational infection
n <- sqrt(nrow(df))
Z <- matrix(m3p$pfitted.values, n, n)
filled.contour(z = Z, x = gluc0, y = linf0,
  xlab = "Glucose (mg/dl)", ylab = "Lymphocytes (%)",
  main = "Probability of post-opererational infection")

par(op)
```

`summary.sback` *Summary for a* `sback` *fitted object*

Description

Takes a fitted object produced by `sback()` and produces various useful summaries from it.

Usage

```
## S3 method for class 'sback'
summary(object, ...)
```

Arguments

`object`	an object of class `sback` as produced by `sback()`.
`...`	other arguments (not implemented)

Value

An object of class `summary.sback` with the information needed to print the results.

Author(s)

Javier Roca-Pardinas, Maria Xose Rodriguez-Alvarez, Stefan Sperlich

See Also

`sback`, `plot.sback`

Examples

```
library(wsbackfit)
set.seed(123)
################################################
# Gaussian Simulated Sample
################################################
set.seed(123)
# Define the data generating process
n <- 1000
x1 <- runif(n)*4-2
x2 <- runif(n)*4-2
x3 <- runif(n)*4-2
x4 <- runif(n)*4-2
x5 <- as.numeric(runif(n)>0.6)

f1 <- 2*sin(2*x1)
f2 <- x2^2
f3 <- 0
f4 <- x4
f5 <- 1.5*x5

mu <- f1 + f2 + f3 + f4 + f5
err <- (0.5 + 0.5*x5)*rnorm(n)
y <- mu + err

df <- data.frame(x1 = x1, x2 = x2, x3 = x3, x4 = x4, x5 = as.factor(x5), y = y)

# Fit the model with a fixed bandwidth for each covariate
m0 <- sback(formula = y ~ x5 + sb(x1, h = 0.1) + sb(x2, h = 0.13)
  + sb(x3, h = 0.1) + sb(x4, h = 0.1), kbin = 30, data = df)

summary(m0)
```

Index