Package 'smof'

July 23, 2025

Type Package

Title Scoring Methodology for Ordered Factors

Version 1.2.2

Date 2024-12-10

Maintainer Adelchi Azzalini <adelchi.azzalini@unipd.it>

Depends R (>= 4.0.0)

Imports stats, methods

Suggests ggplot2, survival, nloptr

ByteCompile yes

NeedsCompilation no

Description Starting from a given object representing a fitted model (within a certain set of model classes) whose (non-)linear predictor includes some ordered factor(s) among the explanatory variables, a new model is constructed and fitted where each named factor is replaced by a single numeric score, suitably chosen so that the new variable produces a fit comparable with the standard methodology based on a set of polynomial contrasts. Two variants of the present approach have been developed, one in each of the next references: Azzalini (2023) <doi:10.1002/sta4.624>, (2024) <doi:10.48550/arXiv.2406.15933>.

License GPL-2 | GPL-3

Encoding UTF-8

Author Adelchi Azzalini [aut, cre] (ORCID: <https://orcid.org/0000-0002-7583-1269>)

Repository CRAN

Date/Publication 2024-12-10 10:40:02 UTC

Contents

`smof-package` *Scoring Methodology for Ordered Factors*

Description

Starting from a given object representing a fitted model (within a certain set of model classes) whose linear predictor includes some ordered factor(s) among the explanatory variables, a new model is constructed and fitted where each named factor is replaced by a single numeric score, suitably chosen so that the new variable produces a fit comparable with the standard methodology based on a set of polynomial contrasts.

Details

The DESCRIPTION file:

Package:	smof
Type:	Package
Title:	Scoring Methodology for Ordered Factors
Version:	1.2.2
Date:	2024-12-10
Authors@R:	person(given = "Adelchi", family = "Azzalini", email = "adelchi.azzalini@unipd.it", role = c("aut", "cre
Maintainer:	Adelchi Azzalini <adelchi.azzalini@unipd.it>
Depends:	R (>= 4.0.0)
Imports:	stats, methods
Suggests:	ggplot2, survival, nloptr
ByteCompile:	yes
NeedsCompilation:	no
Description:	Starting from a given object representing a fitted model (within a certain set of model classes) whose (n
License:	GPL-2 | GPL-3
Encoding:	UTF-8
Author:	Adelchi Azzalini [aut, cre] (<https://orcid.org/0000-0002-7583-1269>)

Index of help topics:

```
print.smof              Methods for 'smof' objects
smof                    Scoring Methodology for Ordered Factors
smof-package            Scoring Methodology for Ordered Factors
smof_refit              Re-fitting an existing 'smof' model for
                        improved optimization
```

Author(s)

Author: Adelchi Azzalini [aut, cre] (<https://orcid.org/0000-0002-7583-1269>) Maintainer: Adelchi Azzalini <adelchi.azzalini@unipd.it>

References

Azzalini, A. (2023). On the use of ordered factors as explanatory variables. *Stat* **12**, e624. doi:10.1002/sta4.624

Examples

```
library(datasets)
data(esoph)
contrasts(esoph$agegp, 2) <- contr.poly(6) # optional
contrasts(esoph$tobgp, 1) <- contr.poly(4) # optional
obj1 <- glm(cbind(ncases, ncontrols) ~ agegp + tobgp + alcgp, family=binomial(), data=esoph)
out0 <- smof(obj1, esoph, "alcgp")
print(summary(out0$object))
```

`smof` *Scoring Methodology for Ordered Factors*

Description

Starting from an object representing a fitted model whose (non-)linear predictor includes some ordered factor(s) among the explanatory variables, a new model is constructed where each named factor is replaced by a single numeric score, suitably chosen so that the new variable produces a data fit comparable with the standard methodology based on a set of polynomial contrasts.

Usage

```
smof(object, data, factors, scoring, fast.fit = FALSE, original = FALSE,
     f.tail=".score", opt.method="Nelder-Mead", opt.control=list(), verbose = 0)
```

Arguments

`object`	an object produced by a fitting function; see 'Details' below for specification of the admissible classes of objects.
`data`	the data frame used for producing `object`.
`factors`	a character vector with the names of the ordered factors of `data` which must be converted to numeric scores.
`scoring`	a list which selects the type of scoring techniques and related ingredients. Its key component is the character string `scoring$type` with possible values `"distr"` and `"spline"`; the other components of the list depend on this value, and are described in the 'Details'. If the argument `scoring` is missing, the default value `list(type="distr", family="gh")` is used.
`fast.fit`	a logical value (default value: `FALSE`) indicating whether a fast-fitting procedure must be used. This option is available only under certain circumstances specified in the 'Details' below.
`original`	a logical value (default value: `FALSE`) indicating whether the original `object` must be included in the returned object.

`f.tail`	a character string representing the suffix to be added to each factor name when the coresponding numeric variable is constructed (default value: `".score"`). It may be suitably adjusted in case the supplied string leads to an invalid variable name.
`opt.method`	a character string with the name of the numerical optimization method, among a set of options. The basic set comprises `"Nelder-Mead"` (default value), `"BFGS"`, `"nlminb"`, all accessible from the **stats** package; the first two choices generate a call to the function `optim`, while `"nlminb"` produces a call to `nlminb`. In addition, if the package **nloptr** is installed on the local system, the set of options is enlarged with `"newuoa"`, `"bobyqa"`, `"cobyla"`, `"sbplx"`.
`opt.control`	a list passed to the optimization function selected by `opt.method` as its `control` argument. It must not be used to turn the minimization problem into a maximization one.
`verbose`	a non-negative integer regulating the amount of messages displayed; it can be 0 (no messages unless necessary, default value), 1 (minimal messaging) or larger than 1 for a more verbose outcome.

Details

In its original formulation, `smof` implements the methodology proposed by Azzalini (2023), briefly summarized in the 'Background' section. It is recommended to read at least that section in case the referenced paper is not examined. The published paper has open access. Later on, since version 1.2.0 of **smof**, a variant methodology has been included, based on the use of splines instead of quantile functions, presented in Azzalini (2024).

Start from an `object` obtained as the outcome from some fitting procedure, whose linear predictor includes one or more ordered factor(s) among the explanatory variables. For each ordered factor whose name is included in vector `factors`, a suitable vector of numeric scores is constructed. The selection process examines the quantiles of the members of a specified parametric class of distributions and selects the member with optimizes (i.e. minimizes) a suitable target criterion. To avoid trivialities, each factor in vector `factors` must have at least three levels.

There are two quite different options to build the numeric scores assigned to an ordered factor. The selection of one of these options is made via the component `type` of the list `scoring`, which can be either `"distr"` or `"spline"`. The other components of `scoring` depends on the chosen `type` and are described below.

If `scoring$type="distr"`, the numeric scores are obtained as quantiles of a probability distribution belonging to a certain parametric family; this route corresponds to the original construction of `smof`, following Azzalini (2023). The admissible parametric families are all obtained by monotonic transformations of a standard normal variate. Specifically, the admissible families and corresponding strings to be specified in `scoring$family` are as follows:

Johnson's S_U	`"SU"`
Tukey's g-and-h	`"gh"`, `"g-and-h"`
Jones and Pewsey's $sinh - arcsinh$	`"sinh-arcsinh"`, `"SAS"`

where either string name can be used when two of them are indicated. All these families involve two parameters for shape regulation; location and scale parameters are not considered, because irrelevant for our purposes. Of the two shape parameters, the first one regulates asymmetry and can

take any value, while the second one regulates tail thickness and must be positive. In each case, the adopted parameterization is the 'standard' one, but explicit specifications are provided in the reference below. The same `family` is employed for all the components of `factors`.

Since version 1.2.2 of the package, the selected quantiles are, in the final stage, linearly mapped to values in the interval $(1, K)$, if K denotes the number of levels of the factor under consideration. This choice is simplifies comparison among different choices of scoring, in a number of ways: first of all, it highlights the difference from the traditional scores $1, 2, \ldots, K$; next, it facilitates comparison among alternative choice of `scorinf$family`; finally, it it facilitates comparison with the scoring produced using spline, to be described shortly, whose scores range between 1 and K. An implication of the present choice of mapping quantiles is that the scores produced now in the example code below differs numerically from the scores displayed in the examples of Azzalini (2023), but the difference is only superficial, since the relative spacings between the old and the new scores are unchanged, and this is the crucial point. However, in case one wants to recover the old type of outcome, this is possible by including the component `scoring` in the call to `smof` and setting `scoring$mapping="none"`.

If `scoring$type="spline"`, the numeric scores are obtained using monotonic spline functions. Specifically, the scores are generated using `splinefun` with `method="monoH.FC"`. Since currently this is the only admissible form of spline, it does not need to be specified. What must be specified is `scoring$in.knots`, the number of internal knots between the fixed extremal knots `1` and `K`, if `K` denotes the number of levels of any given factor. Hence `scoring$in.knots` should be an integer vector with as many components as `factors`; if a shorter vector is supplied, its values will be recycled. Each component of `in.knots` must not exceed the corresponding value of `K-2`. Since each internal knot involves the selection of two numeric values, the total number of fitted parameters equals the sum of twice the `in.knots` values, summed over the components of `factors`.

Estimation of the transformation parameters is performed by a numerical process which optimizes a target criterion which depends on `class(object)`. This numerical scheme is summarized in Section 'Computational aspects'. The admissible classes for `object` are currently as follows, listed along the corresponding target criteria:

class	fitting function (package)	target criterion
lm	lm (stats)	sum of squared residuals
mlm	lm (stats)	[see below]
glm	glm (stats)	deviance
survreg	survreg (survival)	$-$loglikelihood
coxph	coxph (survival)	$-$loglikelihood
coxph.penal	coxph (survival)	$-$loglikelihood
gnm	gnm (gnm)	deviance

For an object of class `mlm`, the target function is formed by summing terms where the contribution from the j-th response variable is $(1 - R_j^2)$, where R_j^2 is the `r-squared` statistic for that component of the fitted model. Note that, in the case of a single response variable, its $(1 - R^2)$ value is equivalent, up to an algebraic transformation, to the sum of squared residuals used for `lm` objects; hence the chosen target criterion for `mlm` models is a direct extension of the one for `lm`'s. The above list of classes may be expanded in the future, depending on feedback.

The rest of this section is slightly of more technical nature, and it may be not of interest to the casual user, especially if the option `fast.fit=TRUE` is not selected. Operationally, estimation of

the scoring$family parameters is performed via optimization of the pertaining target criterion, as indicated by the table above. For each candidate set of parameters, each factor included in factors is replaced by values determined by the quantiles of scoring$family and the current parameters. The name of the new constructed variable is formed by adding .score to the original name. For instance, an ordered factor called ordfac is replaced by the numeric variable ordfac.score both in the linear predictor of object and in the data frame.

If the component scoring$param is not NULL, it is assumed to provide initialization values for the parameter search process. Specifically, if nf denotes the number of elements of factors, a set of nf vectors must be provided, one for each component of factors. In case $scoring$type="distr", scoring$param must be a matrix with dimension (nf,2) where each row vector represents the parameters for the corresponding element of factor; the second columns of this matrix must have positive elements, since they represent tail-weight parameters. In case $scoring$type="smof"), scoring$param must be a list of nf vectors, with lengths 2*in.knots. In this case, it is assumed that each vector comprises two subvectors of ordered values in the range between 1 and M, where M denotes the number of levels of the corresponding factor.

For the numerical fitting procedure, there exist in fact two variants. The more commonly used 'general' variant form is summarized below in the already-mentioned Section 'Computational aspects' below. However, in the prominent cases of an object of class lm or glm, the procedure can be speeded-up by setting fast.fit=TRUE, under certain conditions indicated next. One such condition is that scoring$type="distr" is set. Also, it is required that the fitted model is of a basic form, that is, a model specification via a formula, and a family in the glm case, without non-basic arguments such as offset, subset and alike. If these non-basic arguments are included in the object call, they are ignored for estimation of the scoring$family parameter. However, they are included for producing the final object returned by the function. With this option, the sequence of calls to lm and glm involved by the iterative search procedure is replaced by faster calls to lm.fit and glm.fit. Correspondingly, the internal target function (target.fit) is slightly different from the one used on the more general case (target.gen). Since the selection of the parameters involves an iterative process with dimensionality equal to twice the length of factors and each iteration involves a new data fitting process, the saving in execution time can be appreciable in some cases.

Value

A list with the following components:

call	the calling statement
new.object	an updated version of the original object, with the components of factors in the model replaced by new variables; this object is itself a list, whose structure depends on its class.
new.data	a new data frame where the ordered factors are replaced by numeric variables representing scores.
scoring	a list similar to the input argument with the addition of the estimates of the parameters.
factor.scores	a list of numeric vectors with the scores assigned to the levels of each factor. In case of scores produced by splines, attributes with the spline knots are included.
original.factors	a list with the names and the levels of the original factors.

`target.criterion`
: the final value of the target criterion used for fitting.

`opt`
: the list returned by numerical optimization function, with an additional initial item recording `opt.method`.

Background

The methodology proposed in the reference below deals with the presence of ordered factors used as explanatory variables, hence included in the linear predictor of some model under consideration. For any given ordered factor with *K* levels, say, a set of K numeric scores is introduced, with a certain value assigned to each factor level. In the end, the original factor is effectively replaced by a numeric variable. This scheme represents a refinement of the elementary scoring system based on the basic sequence $1, \ldots, K$, which constitutes a simple time-honoured option to deal with ordered factors, but it is not always appropriate.

There are two variants of the methodology, selected with the value of the component `type` of the list `scoring`. Here we summarize the working of the original formulation, selected by setting `scoring$type="distr"`; this happens implicitly if `scoring` is not specified. The basic logic dealing with the case `scoring$type="spline"` is the same, even if the technique is different. The actual construction of numeric scores proceeds by selecting K quantiles of a distribution belonging to some parametric family. The adoption of a sufficiently flexible parametric family helps to find a scoring system best suited for the data under consideration, hence improving upon the basic sequence $1, \ldots, K$. A concomitant product of this scheme is the identification of numeric values which indicate how the K levels are "really" spaced. Combining these two features, the key feature of the proposal is interpretability of the construction.

The proposed method represents an alternative to the use of polynomial contrasts, which is the default action taken by R for ordered factors; see the documentation of `contr.poly`.

In the proposed logic, the constructed scores are intended to be used, and interpreted, without further manipulation. Hence, for instance, building a polynomial form using one such variable would diverge somewhat from the proposed logic, although still conceivable. With a single numeric variable to represent a given factor, one cannot expect to achieve the same numerical fit to the data as obtained the polynomial contrasts built for the original factor, when these constrasts involve high degrees polynomials, and correspondingly several parameters. However, a range of numerical explorations has indicated that in many cases the resulting fit is equal or similar to the one achieved via polynomial constrasts, with non-negligible simplification in the model specification, and easier interpretation,

In a nutshell, the aim of the approach is to achieve a satisfactory data fit while improving an model parsimony, with simple interpretability of the score system.

The alternative variant of the methodology is selected by setting `scoring$type="spline"`. The underlying principle is similar to the one just described, but it makes use of splines instead of quantile functions. Its operational working is described in the 'Details'.

For a more comprehensive exposition and discussion, see the references below.

Computational aspects

The fitting step for selecting the transformation(s) parameters proceeds by an iterative scheme, whose essence is as follows. For each choice of the transformation parameter(s), a call to `update` of the `object` provided, using suitably modified linear predictor and data, delivers a new fitting,

with attached a corresponding value of the target criterion. For this process, work parameters are introduced which are free from the constraints implicit in either variant of the procedure, that is, either with `scoring$type="distr"` or with `scoring$type="spline"`. An iterative optimization process of the target criterion (using the selected `opt.method`) leads to optimized work parameters. In the final stage, the selected work parameters are mapped to actual model-meaningful parameters which identify a corresponding fitted model.

In some cases, the optimization step of `smof` can run into problems, typically when many parameters regulating the numeric scores are involved. This situation may be flagged by the warning message

```
Possibly unsatisfactory outcome from optimization function.
```

In such a case, further information is printed even if `verbose=0`. Since this information simply replicates what is delivered by the optimization function selected with `opt.method`, the documentation of that function must be examined to decipher the meaning of the message(s); sometimes, this may not be entirely obvious.

To handle these events, sometimes it suffices to increase the number of attempted iterations via `opt.control` and re-run the `smof`. Another simple possibility is to change the selected `opt.method`. With more awkward situations, a more elaborate use of `opt.control` is required. Alternatively, consider using of the function `smof_refit`.

Note

For subsequent computations on the object returned by `smof`, difficulties may arise if the call to the fitting function does not set `model=TRUE`. This is not a problem with `lm` and `glm`, if their default setting `model=TRUE` has not been modified. The default setting of `coxph` is instead `model=FALSE`. This implies, for instance, that issuing the `survival` command `survfit(smof4$new.object)`, right after running the code of Example 4 below, would cause an error. The main route to avoid this issue is to set `model=TRUE` in the call to the fitting function, that is, `coxph` or whatever function is used. Alternatively, if one does not want to refit an already existing object, there exist various ways to overcome this snag; the simplest one is to write

```
new.data <- smof4$new.data
s <- survfit(smof4$new.object)
```

This indication is temporary and it may be superseded by a different design in future versions of the package.

Author(s)

Adelchi Azzalini

References

Azzalini, A. (2023). On the use of ordered factors as explanatory variables. *Stat* **12**, e624. doi:10.1002/sta4.624

Azzalini, A. (2024). On the use of splines for representing ordered factors. *arXiv:2406.15933*, doi:10.48550/arXiv.2406.15933

See Also

smof_refit, nlminb, optim, contr.poly, update, splinefun, lm, lm.fit, glm, glm.fit

Examples

```
# Example 1, reconstructs Tables 1 and 2 (second part) of the reference
message("--- Example 1: esoph data ---")
library(datasets)
data(esoph)
contrasts(esoph$agegp, 2) <- contr.poly(6)
contrasts(esoph$tobgp, 1) <- contr.poly(4)
fit1 <- glm(cbind(ncases, ncontrols) ~ agegp + tobgp + alcgp, family=binomial(), data=esoph)
message("original fit:")
print(summary(fit1))
smof1 <- smof(fit1, esoph, "alcgp")
# to select the Johnson's S_U family of distributions, write instead:
# smof1 <- smof(fit1, esoph, "alcgp", scoring=list(type="distr", family="SU"))
print(smof1)
print(summary(smof1))
plot(smof1, type="b", pch=20, col=4)
#
# Example 2 , reconstructs Tables 3 and 4 (first part) of the reference
if(require(ggplot2, quietly=TRUE)) {
message("--- Example 2: diamonds data ---")
data(diamonds, package="ggplot2")
dmd <- data.frame(diamonds[seq(1, 53940, by=100),]) # use a subset of the data
dmd <- dmd[-c(518, 519, 523),] # remove three outliers
contrasts(dmd$clarity, 3) <- contr.poly(8)
contrasts(dmd$color, 4) <- contr.poly(7)
contrasts(dmd$cut, 1) <- contr.poly(5)
fit2 <- lm(sqrt(price) ~ carat + clarity + color + cut, data=dmd)
smof2 <- smof(fit2, dmd,  c("color", "clarity"))
message("smof fit:")
print(smof2)
print(summary(smof2))
plot(smof2, which="clarity")
} # end diamonds example
#
# Example 3
if(require(survival, quietly=TRUE)) {
message("--- Example 3: lung data ---")
lung0 <- lung
lung0$ph.karno <- ordered(lung0$ph.karno)
contrasts(lung0$ph.karno, 3) <- contr.poly(6)
fit3 <- survreg(Surv(time, status) ~ ph.karno, data=lung0)
smof3 <- smof(fit3, lung0, "ph.karno")
print(summary(smof3))
plot(smof3)  # Karnofsky scores do not seem to be linearly spaced
#
message("--- Example 4: PBC data ---")
data(pbc, package="survival")
pbc$stage <- ordered(pbc$stage)
```

```
fit4 <- coxph(Surv(time) ~ strata(status) + stage, data=pbc)
smof4 <- smof(fit4, data=pbc, factors="stage")
print(summary(smof4))
plot(smof4)
} # end of survival examples
```

---

smof-methods *Methods for* smof *objects*

Description

The list of methods that apply to smof objects

Usage

```
## S3 method for class 'smof'
print(x, ...)
## S3 method for class 'smof'
plot(x, which, ...)
## S3 method for class 'smof'
summary(object,  ...)
## S3 method for class 'summary.smof'
print(x, ...)
## S3 method for class 'smof'
predict(object, newdata, ...)
## S3 method for class 'smof'
coef(object, complete=FALSE, ...)
## S3 method for class 'smof'
anova(object, ...)
```

Arguments

object	an object returned by smof.
x	an object returned by smof, except for print.summary.smof where x is the outcome of summary.smof.
which	either a vector or a two-elements list; see 'Details' for full specification.
newdata	a data frame which includes ordered factors with the same names and levels as those in the data frame used to produce object; see 'Details' for additional information.
complete	logical value (default: FALSE); when TRUE, the coefficients of the fitted model are included after the parameters of the scoring transformation.
...	arguments passed through to other methods. For anova it is ignored.

Details

There are two main parts in the outcome of `summary.smof`. One is the outcome of the selection of the factor(s) transformation(s); the other part, denoted `Final fitting call`, represents the newly fitted model with the original ordered factors replaced by numeric scores. The corresponding `data.frame`, denoted `new.data`, is one of the components of the object returned by `smof`.

If `which` is a character vector, its components are interpreted as names of the `factors` in the calling statement of the object to be plotted, producing a set of graphs where the numeric scores of each named factor are plotted versus the equally spaced ticks associated to the original level names. The same effect is obtained when `which` is a numeric vector of integers, which then select the corresponding components of the `factors` sequence.

If `which` is a list, its first element is assumed to be a vector having the meaning just described. After the pertaining set of graphs has been completed, the second element of the list is passed to the plotting method for the object produced by the model fitting procedure. Currently this option operates only for objects which inherits from class `lm`; specifically, it works for objects originated by a call to `lm` or to `glm`.

With `predict.smof`, the outcome of a `smof` fit is applied to new data frame which includes ordered factors analogous to those used to compute the `smof` transformation. Only the factors of `newdata` with the same name as those processed by `smof` to produce `object` are examined. The levels of these factors must coincide with or be a subset of those of the original data frame.

Note the difference between the role played by `predict.smof` and the one of `predict` methods for most other classes. Usually the returned values pertain to the response variable, or to some related entity, while here the outcome refers to explanatory variables.

Value

For `summary.smof`, a list of class `summary.smof`. For `predict.smof`, a data frame. For `plot.smof`, `NULL` with graphical side effects.

Author(s)

Adelchi Azzalini

See Also

`smof`, `lm`, `glm`

Examples

```
library(datasets)
data(esoph)
contrasts(esoph$agegp, 2) <- contr.poly(6)
contrasts(esoph$tobgp, 1) <- contr.poly(4)
fit1 <- glm(cbind(ncases, ncontrols) ~ agegp + tobgp + alcgp, family=binomial(), data=esoph)
smof1 <- smof(fit1, esoph, "alcgp")
print(smof1)
print(summary(smof1))
plot(smof1, type="b", pch=19, col="blue")
plot(smof1, which=list(1, 1:4))
predict(smof1, newdata=esoph[seq(1, 88, by=8), ])
```

smof_refit *Re-fitting an existing* smof *model for improved optimization*

Description

Given a model fitted by smof, this function helps to improve the achieved fitting level (represented by the value of the target criterion) by launching new numerical optimizations of the underlying target function. The search process is initiated with parameters randomly chosen in the vicinity of those of the object provided.

Usage

```
smof_refit(object, searches = 10, sd = 1, opt.control=list(), verbose = 1)
```

Arguments

object	an object returned by a call to smof; that call must include the argument original=TRUE.
searches	a positive integer representing the number of attempted searches.
sd	the standard deviation of the zero-mean normal variates used for earch search to generate the initial set of parameters from which to initiate a numerical optimization.
opt.control	a list passed to the optimization function (the same used to create object) as its control argument. It must not be used to turn the minimization problem into a maximization one.
verbose	an integer regulating the amount of messages displayed; it can be 0 (no messages), 1 (default value) or larger than 1 for a more verbose outcome.

Details

Section 'Computational aspects' of the smof documentation provides some simple suggestions to overcome numerical difficulties arising in the fitting process. For harder problems, smof_refit aims at improving the results obtained by an initial call to smof via a sequence of calls to smof itself, each time starting from the last best fit obtained up to that point.

In 'regular' situations, it is expected that the main usage of this function is to improve the fitting of models produced with scoring$type="spline", since this option typically generates a model with more parameters than an equivalent model with scoring$type="distr", hence generally more problematic at the optimization step.

Since the initial values of each optimization step are randomly generated (in the viciniy of the best known parameter set), it is advisable to make a preliminary call to set.seed, to ensure replicability of the results.

Value

an object of class smof

See Also

smof, set.seed

Examples

```
library(datasets)
data(esoph)
fit <- glm(cbind(ncases, ncontrols) ~ agegp + tobgp + alcgp, family=binomial(), data=esoph)
smof2 <- smof(fit, esoph, c("agegp", "alcgp"),
              scoring=list(type="spline", in.knots=c(2,1)), original=TRUE)
set.seed(1)
smof2a <- smof_refit(smof2, searches=4, opt.control=list(maxit=50))
# smof2b <- smof_refit(smof2a)  # further improvement can be attempted
```

Index