Journal of Statistical Software

June 2014, Volume 58, Issue 5. *http://www.jstatsoft.org/*

Smoothing Spline ANOVA Models: R Package gss

Chong Gu
Purdue University

Abstract

This document provides a brief introduction to the R package **gss** for nonparametric statistical modeling in a variety of problem settings including regression, density estimation, and hazard estimation. Functional ANOVA (analysis of variance) decompositions are built into models on product domains, and modeling and inferential tools are provided for tasks such as interval estimates, the "testing" of negligible model terms, the handling of correlated data, etc. The methodological background is outlined, and data analysis is illustrated using real-data examples.

Keywords: Bayesian confidence interval, cross-validation, functional ANOVA decomposition, Kullback-Leibler projection, mixed-effect models, penalized likelihood.

1. Introduction

Nonparametric function estimation using stochastic data, otherwise known as smoothing, has been studied by generations of statisticians. While scores of methods have proved successful for univariate smoothing, ones practical in multivariate settings number far less. Smoothing spline ANOVA models are a versatile family of smoothing methods that are suitable for both univariate and multivariate problems.

In this article, we introduce the package **gss** for R (R Core Team 2014) that embodies suites of functions implementing smoothing spline ANOVA models in the settings of Gaussian and non-Gaussian regression, density estimation, and hazard estimation. The first public release of **gss** dated back to 1999, when the total number of R packages on CRAN, the Comprehensive R Archive Network, was in dozens. The package was originally designed as a front end to **RKPACK** (Gu 1989), a collection of Ratfor routines for Gaussian regression. Over the years, new functionalities have been added, numerical efficiency improved, the user interface refined, and **gss** has now matured into a comprehensive package that can be used in a great variety of problem settings. As active development tapers off, **gss** is likely to remain stable in the foreseeable future, and it is time to compile an introductory document for the current version of the package.

A treatise on the theory and practice of smoothing spline ANOVA models can be found in a recently updated monograph by the author (Gu 2013), which contains detailed discussions of **gss** with extensive code illustrations. This document is intended as a software-oriented executive summary of the monograph, and for the exposition of the key methodological ingredients, technical discussions are unavoidable. Attempts have been made to keep the technical discussions lucid if not rigorous, yet readers unfamiliar with the methodology may still find the contents more mathematical than they might prefer.

The rest of the article is organized as follows. In Section 2, the methodological background is outlined, and in Section 3, the basic functionality of the **gss** suites are introduced with limited illustrations. Model configurations via explicit and implicit means are described in Section 4, and extra features such as semiparametric models and mixed-effect models are discussed in Section 5. After a brief discussion of the numerical engine in Section 6, three real-data examples are presented in Section 7 to demonstrate data analysis using **gss** facilities. Section 8 notes on some further software utilities without detailed elaboration, and Section 9 concludes the article with a brief summary.

Chunks of executable R code are collected in the supplementary material. The output of the code, which includes a few figures, are however not reproduced in this document.

2. Methodological background

Observing $Y_i \sim \mathcal{N}(\eta(x_i), \sigma^2)$, $i = 1, \ldots, n$ on $x_i \in [0, 1]$, one may estimate $\eta(x)$ via the minimization of a penalized least squares functional,

$$\frac{1}{n}\sum_{i=1}^{n}\big(Y_i - \eta(x_i)\big)^2 + \lambda \int_0^1 \big(\eta''(x)\big)^2 dx. \tag{1}$$

The minimization takes place in a function space $\big\{f : \int_0^1 \big(f''(x)\big)^2 dx < \infty\big\}$ of infinite dimension, and the solution is known as a smoothing spline. The minimizer of (1) is a piecewise cubic polynomial, so it is called a cubic smoothing spline.

The method in (1) has been generalized from univariate Gaussian regression to a variety of estimation problems on generic domains, which include Gaussian and non-Gaussian regression, density estimation, and hazard estimation. Key ingredients of the methodology are outlined below.

Smoothing splines are not to be confused with penalized regression splines, with which esimation takes place in a finite-dimensional function space pre-configured using basis functions of local support. Penalized regression splines are numerically more vulnerable to the curse of dimensionality, as the number of local-support basis functions explodes when the dimension of the domain goes up.

2.1. Penalized likelihood estimation

The penalized least squares functional in (1) is a special case of the general penalized likelihood functional,

$$L(\eta) + \lambda\, J(\eta), \tag{2}$$

where $L(\eta)$ is usually taken as the minus log likelihood of the observed data and $J(\eta)$ is a quadratic "roughness" functional; the minimization of (2) takes place in $\mathcal{H} = \big\{f : J(f) < \infty\big\}$. With different configurations of $L(\eta)$ and $J(\eta)$, the minimizer of (2) may or may not be a piecewise polynomial, but it is called a smoothing spline for being the solution to a penalized minimization problem.

Function evaluation $\eta(x)$ appears in $L(\eta)$, so the evaluation functional $[x]f = f(x)$ should be continuous in $\mathcal{H} = \big\{f : J(f) < \infty\big\}$. A space in which the evaluation functional is continuous is known as a reproducing kernel Hilbert space possessing a reproducing kernel $R(x, y)$, a nonnegative-definite function satisfying the reproducing property, $\big(f, R(x, \cdot)\big) = f(x), \forall f \in \mathcal{H}$, where $(\cdot, \cdot)$ is the inner product in $\mathcal{H}$.

$J(f)$ is usually a square semi norm in $\mathcal{H}$, and a square full norm in an orthogonal complement of the null space $\mathcal{N}_J = \big\{f : J(f) = 0\big\}$ of $J(f)$, $\mathcal{H}_J = \mathcal{H} \ominus \mathcal{N}_J$. The reproducing kernel $R_J(x, y)$ of $\mathcal{H}_J$ satisfies $J\big(R_J(x, \cdot), f\big) = f(x)$, $\forall f \in \mathcal{H}_J$. When $L(\eta)$ depends on η only through $\eta(x_i)$, $i = 1, \ldots, n$, as is the case in (1), the minimizer of (2) resides in the space $\mathcal{N}_J \oplus \mathrm{span}\big\{R_J(x_i, \cdot), i = 1, \ldots, n\big\}$, of finite dimension (Kimeldorf and Wahba 1971).

For $J(f) = \int_0^1 \big(f''(x)\big)^2 dx$ as in (1), $\mathcal{N}_J = \mathrm{span}\{1, x\}$, and an orthogonal complement of $\mathcal{N}_J$ is $\mathcal{H}_J = \big\{f : \int_0^1 f(x)dx = \int_0^1 f'(x)dx = 0, \int_0^1 \big(f''(x)\big)^2 dx < \infty\big\}$.

2.2. Functional ANOVA decomposition

On $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$, one has a functional ANOVA decomposition of $\eta(x) = \eta(x_{\langle 1\rangle}, x_{\langle 2\rangle})$,

$$\begin{aligned}\eta(x) &= (I - A_1 + A_1)(I - A_2 + A_2)\eta \\ &= A_1A_2\eta + (I - A_1)A_2\eta + A_1(I - A_2)\eta + (I - A_1)(I - A_2)\eta \\ &= \eta_\emptyset + \eta_1(x_{\langle 1\rangle}) + \eta_2(x_{\langle 2\rangle}) + \eta_{12}(x_{\langle 1\rangle}, x_{\langle 2\rangle}),\end{aligned} \tag{3}$$

where I is the identity operator, A_1, A_2 are averaging operators acting respectively on arguments $x_{\langle 1\rangle}$, $x_{\langle 2\rangle}$ that satisfy $A1 = 1$; subscripts in brackets denote coordinates of a point on a multi-dimensional domain while ordinary subscripts are reserved for multiple points. Examples of averaging operators include $Af = \int_a^b f(x)dx/(b - a)$ and $Af = \sum_{i=1}^m f(x_i)/m$. The two-way ANOVA decomposition of (3) also implies a one-way ANOVA decomposition on $\mathcal{X}$ with $A = A_1A_2$, $\eta(x) = (I - A + A)\eta = A\eta + (I - A)\eta = \eta_\emptyset + \eta_x(x)$, where $\eta_x = \eta_1 + \eta_2 + \eta_{12}$ for η_1, η_2, and η_{12} as in (3). Similar constructions in more than two dimensions can be done directly or recursively.

For $\mathcal{X}_1 \times \mathcal{X}_2$ discrete, $\eta(x_{\langle 1\rangle}, x_{\langle 2\rangle})$ is a matrix of treatment means usually denoted by μ_{ij} in standard ANOVA model notation, with (3) in the form of

$$\begin{aligned}\mu_{ij} &= \mu_{\cdot\cdot} + (\mu_{i\cdot} - \mu_{\cdot\cdot}) + (\mu_{\cdot j} - \mu_{\cdot\cdot}) + (\mu_{ij} - \mu_{i\cdot} - \mu_{\cdot j} + \mu_{ii}) \\ &= \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij},\end{aligned}$$

where $\mu_{i\cdot} = \sum_j c_j\mu_{ij}$ for $\sum_j c_j = 1$, $\mu_{\cdot j} = \sum_i d_i\mu_{ij}$ for $\sum_i d_i = 1$, and $\mu_{\cdot\cdot} = \sum_{i,j} c_jd_i\mu_{ij}$.

ANOVA decompositions can be built into (2) via the configuration of $J(\eta)$ in the so-called tensor product reproducing kernel Hilbert spaces, and the resulting estimates are called tensor product splines. Schematically, one may write the ANOVA decomposition as $\eta = \sum_\beta \eta_\beta$, with $J(\eta)$ of the form $J(\eta) = \sum_\beta \theta_\beta^{-1} J_\beta(\eta_\beta)$, where $J_\beta(\eta_\beta)$ quantify the roughness of η_β and θ_β are tuning parameters adjusting their relative weights in $J(\eta)$.

Selective term elimination in ANOVA decompositions helps to combat the curse of dimensionality in estimation; it also facilitates the interpretation of the fitted models. Models containing only main effects are known as additive models (Hastie and Tibshirani 1990).

2.3. Efficient approximation

As noted earlier, the minimizer of (2) often resides in $\mathcal{N}_J \oplus \mathrm{span}\{R_J(x_i,\cdot), i=1,\ldots,n\}$, permitting numerical computation, but the computation is generally of the order $O(n^3)$. When $L(\eta)$ depends on η via more than a finite number of function evaluations, however, the exact solution is usually intractable numerically.

Under mild conditions, as $n \to \infty$ and $\lambda \to 0$, the minimizer $\hat\eta$ of (2) in $\mathcal{H}$ converges to the true η_0 at a rate $U(\hat\eta - \eta_0) = O_p(n^{-1}\lambda^{-1/r} + \lambda^p)$, where $U(\hat\eta-\eta_0)$ is a setting-specific quadratic loss, $r > 1$ codes the "smoothness" of functions in $\mathcal{H}$ implied by $J(f)$, and $p \in [1,2]$ depending on how smooth η_0 is; in the setting of (1), $U(g) = \int_0^1 g^2(x)f(x)dx$ for $f(x)$ the limiting density of $\{x_i\}$, $r=4$, and $p=2$ when $\int_0^1 \big(\eta_0^{(4)}(x)\big)^2 dx < \infty$. Now consider a space

$$\mathcal{H}^* = \mathcal{N}_J \oplus \mathrm{span}\{R_J(z_j,\cdot), j=1,\ldots,q\}, \tag{4}$$

where $\{z_j\}$ is a random subset of $\{x_i\}$. The minimizer $\hat\eta^*$ of (2) in $\mathcal{H}^*$ converges to η_0 at the same rate as $\hat\eta$ when $q\lambda^{2/r} \to \infty$, hence is an efficient approximation of $\hat\eta$. The optimal convergence rate $O_p(n^{-pr/(pr+1)})$ is achieved at $\lambda \asymp n^{-r/(pr+1)}$, so one needs $q \asymp n^{2/(pr+1)+\epsilon}$, $\forall\epsilon > 0$. One may calculate $\hat\eta^*$ for practical estimation, as is done in most of the **gss** suites, and the computation is of the order $O(nq^2)$.

Unlike penalized regression splines, the space $\mathcal{H}^*$ is data-adaptive rather than pre-configured, with resources allocated only where needed. This does not make the estimation problem any easier, but does make the numerical tasks much more manageable on high-dimensional domains.

Unless $q = n$, the minimizer of (1) in $\mathcal{H}^*$ is no longer a piecewise cubic polynomial, but we will abuse the terminology and still call it a cubic spline.

2.4. Cross-validation

For the method in (2) to work in practice, a critical issue is the proper selection of smoothing parameters, the λ in front of $J(\eta)$ and the θ's hidden in $J(\eta)$ for tensor product splines.

For Gaussian regression as in (1), write $\mathbf{Y} = \big(Y_1,\ldots,Y_n\big)^\top$ and $\hat{\mathbf{Y}} = \big(\eta_\lambda(x_1),\ldots,\eta_\lambda(x_n)\big)^\top$, where the dependence of $\hat\eta = \eta_\lambda$ on λ is spelled out. One has $\hat{\mathbf{Y}} = A(\lambda)\mathbf{Y}$, where $A(\lambda)$ is the so-called smoothing matrix and the argument λ also represents the θ's if present. One may select the smoothing parameters by minimizing the generalized cross-validation score of Craven and Wahba (1979),

$$V(\lambda) = \frac{n^{-1}\mathbf{Y}^\top\big(I - A(\lambda)\big)^2\mathbf{Y}}{\big\{n^{-1}\mathrm{tr}\big(I - \alpha A(\lambda)\big)\big\}^2}, \tag{5}$$

for $\alpha = 1$, which is designed to track the mean square error $\sum_{i=1}^n \big(\eta_\lambda(x_i) - \eta_0(x_i)\big)^2$. Cross-validation occasionally yields severe undersmoothing, and a fudge factor $\alpha = 1.4$ proves to be effective in curbing undersmoothings on "bad" samples while maintaining the generally favorable performances of cross-validation on "good" ones.

In most other settings, one may use a setting-specific Kullback-Leibler discrepancy $\mathrm{KL}(\eta_0, \eta_\lambda)$ as the performance measure for the minimizer η_λ of (2), and schematically, $\mathrm{KL}(\eta_0, \eta_\lambda) = A(\eta_\lambda)+B(\eta_0, \eta_\lambda)+C(\eta_0)$, where $A(\eta_\lambda)$ can be computed, $C(\eta_0)$ can be dropped, and $B(\eta_0, \eta_\lambda)$ can be estimated by some version of cross-validation. The resulting cross-validation scores are typically of the form

$$V(\lambda) = A(\eta_\lambda) + \hat{B}(\eta_0, \eta_\lambda) = L(\eta_\lambda) + \alpha P(\eta_\lambda) \tag{6}$$

for $\alpha = 1$, where the likelihood $L(\eta_\lambda)$ decreases with λ while $P(\eta_\lambda)$ moves in the opposite direction; see, e.g., Gu and Wang (2003). A fudge factor $\alpha > 1$ may improve performance in some settings but not in some others, though a larger α generally yields a smoother estimate.

2.5. Bayesian confidence intervals

The square norm $J(f)$ in $\mathcal{H}_J$ and the reproducing kernel $R_J(\cdot,\cdot)$ are "inverses" of each other, and $\lambda J(\eta)$ in (1) acts like the minus log likelihood of a Gaussian process prior with $\mathsf{E}\big[\eta(x)\big] = 0$ and $\mathsf{E}\big[\eta(x)\eta(y)\big] \propto R_J(x, y)$. The minimizer $\hat{\eta}$ yields the posterior mean under such a prior and one may also calculate the posterior standard deviation $s(x)$, yielding Bayesian confidence intervals, $\hat{\eta}(x) \pm 1.96\, s(x)$, for $\eta(x)$ (Wahba 1983). Doing the analysis on individual terms in an ANOVA decomposition, one gets the component-wise intervals (Gu and Wahba 1993).

When $L(\eta)$ is non-quadratic such as with non-Gaussian regression, one may consider its quadratic approximation at $\hat{\eta}$, $Q_{\hat{\eta}}(\eta)$; the approximate posterior minus log likelihood $Q_{\hat{\eta}}(\eta) + \lambda J(\eta)$ is Gaussian with mean $\hat{\eta}$, based on which approximate Bayesian confidence intervals can be constructed.

2.6. Kullback-Leibler projection

ANOVA structures may be enforced via selective term elimination in estimation, and may be inferred from fitted models containing possibly redundant terms. The latter task resembles hypothesis testing, with $H_0 : \eta \in \mathcal{H}_0$ versus $H_a : \eta \in \mathcal{H}_0 \oplus \mathcal{H}_1$; for an example, consider $\mathcal{H}_0 = \{\eta : \eta = \eta_\emptyset + \eta_1 + \eta_2\}$ and $\mathcal{H}_1 = \{\eta : \eta = \eta_{12}\}$.

Lacking sampling distributions in settings with infinite-dimensional nulls, the classical testing approach is of little help in this situation. Instead, an approach based on the Kullback-Leibler geometry was developed in Gu (2004): one calculates an estimate $\hat{\eta} \in \mathcal{H}_0 \oplus \mathcal{H}_1$, obtains its Kullback-Leibler projection $\tilde{\eta} \in \mathcal{H}_0$ by minimizing a setting-specific $\mathrm{KL}(\hat{\eta}, \eta)$ over $\eta \in \mathcal{H}_0$, then inspects an "entropy decomposition," $\mathrm{KL}(\hat{\eta}, \eta_c) = \mathrm{KL}(\hat{\eta}, \tilde{\eta}) + \mathrm{KL}(\tilde{\eta}, \eta_c)$, an exact or approximate identity, where η_c is a degenerate fit such as a constant regression function or a uniform density. When $\mathrm{KL}(\hat{\eta}, \tilde{\eta})/\mathrm{KL}(\hat{\eta}, \eta_c)$ is small, one loses little by cutting out $\mathcal{H}_1$.

3. Basic functionality: Estimation and inference

Listed in Table 1 are **gss** suites with brief descriptions. To be presented in this section are their basic user interfaces with core arguments and some defaults. The syntax of the **gss** suites has been designed to resemble that of the `lm` and `glm` suites in base R. Each suite has a fitting function and utility functions for the evaluation of the fits, and most also have a method `project` implementing the Kullback-Leibler projection or a variant thereof.

Suite	Purpose/Feature
`ssanova`	Gaussian regression
`ssanova9`	Gaussian regression, with correlated data
`ssanova0`	Gaussian regression, legacy routine
`gssanova`	Non-Gaussian regression, direct cross-validation
`gssanova1`	Non-Gaussian regression, indirect cross-validation
`gssanova0`	Non-Gaussian regression, legacy routine with indirect CV
`ssden`	Density estimation
`ssden1`	Density estimation, in high dimensions
`sscden`	Conditional density estimation
`sscden1`	Conditional density estimation, for large samples
`ssllrm`	Conditional density estimation, with discrete y in $f(y\|x)$
`sshzd`	Hazard estimation, with or without covariate
`sshzd1`	Hazard estimation, with continuous covariate
`sscox`	Estimation of relative risk in proportional hazard models

Table 1: **gss** suites with brief descriptions.

3.1. Regression suites

For Gaussian regression, one may use `ssanova` or `ssanova0`. For non-Gaussian regression, one may use `gssanova`, `gssanova1`, or `gssanova0`. For Gaussian regression with correlated data, one may use `ssanova9`.

`ssanova`

The following sequence loads the NOx data included in **gss** and calculates the minimizer of (1) in $\mathcal{H}^*$ of (4), with λ minimizing $V(\lambda)$ in (5) for $\alpha = 1.4$.

```
R> data("nox", package = "gss")
R> fit <- ssanova(log(nox) ~ equi, data = nox)
```

The default $\alpha = 1.4$ is based on simulations of limited scales (Kim and Gu 2004), and may be overridden via an argument `alpha`. One may also choose to select λ using the GML score of Wahba (1985) through `method = "m"`. To evaluate the fit on a grid, one may try

```
R> xx <- sort(nox$equi)
R> est <- predict(fit, data.frame(equi = xx), se = TRUE)
```

where `est` is a list with $\hat{\eta}(x)$ in `est$fit` and $s(x)$ in `est$se`. The fitted curve can then be plotted along with the Bayesian confidence intervals.

```
R> plot(nox$equi, log(nox$nox), col = 3)
R> lines(xx, est$fit)
R> lines(xx, est$fit + 1.96 * est$se, col = 5)
R> lines(xx, est$fit - 1.96 * est$se, col = 5)
```

The default q in (4) is set at $10n^{2/9}$, again based on simulations (Kim and Gu 2004), and can be overridden via an argument `nbasis`. Due to the random selection of $\{z_j\}$ in (4), repeated calls to `ssanova` generally return slightly different fits unless $q = n$, but one may ensure the same selection of $\{z_j\}$ by using an argument `seed`, or one may pass the same $\{z_j\}$ into subsequent calls through `id.basis = fit$id.basis`.

To fit a tensor product spline on $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$ with all terms included, use

```
R> fit <- ssanova(y ~ x1 * x2)
```

and to evaluate $\eta_1(x) + \eta_{12}(x)$ for $x_{\langle 1 \rangle}$ in `xx1` and $x_{\langle 2 \rangle}$ in `xx2`, say, with standard errors, try

```
R> predict(fit, data.frame(x1 = xx1, x2 = xx2), se = TRUE,
+    inc = c("x1", "x1:x2"))
```

To calculate the Kullback-Leibler projection into the space of additive models, use

```
R> project(fit, inc = c("x1", "x2"))
```

which returns a list object with the element `ratio` containing the ratio $\mathrm{KL}(\hat{\eta}, \tilde{\eta})/\mathrm{KL}(\hat{\eta}, \eta_c)$. To fit an additive model, use

```
R> fit <- ssanova(y ~ x1 + x2)
```

`gssanova` *and* `gssanova1`

Non-Gaussian regression models can be fitted using `gssanova` or `gssanova1`, which calculate the minimizer of (2) in $\mathcal{H}^*$ of (4); `gssanova` uses direct cross-validation as in (6) for smoothing parameter selection, while `gssanova1` employs the indirect cross-validation of Gu (1992a). Discussions concerning direct and indirect cross-validation can be found in Gu (2013, Section 5.2). Practical performances of `gssanova` and `gssanova1` for different distribution families were compared in Gu (2013, Section 5.4, 8.6) via limited simulations.

The syntax is similar to that of `glm` in base R, but the argument `family` only takes character strings from the list `"binomial"`, `"poisson"`, `"Gamma"`, `"inverse.gaussian"`, `"nbinomial"`, `"weibull"`, `"lognorm"`, and `"loglogis"`; only one link is used for each family, one that is free of constraint. The likelihood term $L(\eta)$ in (2) is of the form,

$$L(\eta) = \frac{1}{n} \sum_{i=1}^{n} l_i\big(\eta(x_i); Y_i\big).$$

The link and the minus log likelihood $l(\eta; y)$ for exponential families binomial, Poisson, Gamma, and inverse Gaussian are listed in Table 2, along with those of the negative binomial family. For the binomial family, the response may be two columns of $(y_i, m_i - y_i)$, or a column of y_i/m_i with m_i entered via an optional argument `weights`. For the negative binomial family, the response is either two columns of (y_i, ν_i), or a column of y_i with a common ν to be estimated along with $\eta(x)$.

The following sequence generates some synthetic binomial data and calculates a cubic spline logistic fit.

Family	Response density $f(y)$	Link	$l(\eta; y) = -\log f(y)$
Binomial	$\binom{m}{y} p^y (1-p)^{m-y}$	$\eta = \log\{p/(1-p)\}$	$-y\eta + m\log(1+e^\eta)$
Poisson	$\lambda^y e^{-\lambda}/y!$	$\eta = \log\lambda$	$-y\eta + e^\eta$
Gamma	$\frac{1}{\beta^\alpha \Gamma(\alpha)} y^{\alpha-1} e^{-y/\beta}$	$\eta = \log\mu = \log(\alpha\beta)$	$ye^{-\eta} + \eta$
Inv. Gauss.	$\frac{1}{\sqrt{2\pi\sigma^2}} y^{-3/2} e^{-(y-\mu)^2/2\sigma^2\mu^2 y}$	$\eta = \log\mu$	$ye^{-2\eta}/2 - e^{-\eta}$
Neg. Binom.	$\frac{\Gamma(\nu+y)}{y!\,\Gamma(\nu)} p^\nu (1-p)^y$	$\eta = \log\{p/(1-p)\}$	$(\nu+y)\log(1+e^\eta) - \nu\eta$

Table 2: Likelihood for exponential families in `gssanova`.

```
R> x <- (0:100)/100
R> eta <- 400 * x^5 * (1 - x)^3 - 1
R> p <- plogis(eta)
R> y <- rbinom(x, 3, p)
R> fit <- gssanova(cbind(y, 3 - y) ~ x, family = "binomial")
```

`gssanova` fits can also be evaluated using `predict`, with $\hat{\eta}(x)$ and $s(x)$ on the link scale.

```
R> est <- predict(fit, data.frame(x = x), se = TRUE)
```

The fit can then be plotted along with Bayesian confidence intervals on the probability scale.

```
R> plot(x, plogis(est$fit), type = "l", ylim = c(0, 1))
R> points(x, y/3, col = 3)
R> lines(x, plogis(est$fit + 1.96 * est$se), col = 5)
R> lines(x, plogis(est$fit - 1.96 * est$se), col = 5)
```

The method `project` also works with `gssanova` fits.

The Weibull, log normal, and log logistic families are the same accelerated life models as implemented in the `survreg` suite of the **survival** package (Therneau and Grambsch 2000; Therneau 2014). The response is of the form $Y = (X, \delta, Z)$, where $X = \min(T, C)$ is the follow-up time for T the lifetime of an item and C the right-censoring time, $\delta = I_{[T \le C]}$ is the censoring indicator, and $Z < X$ is a possible left-truncation time at which the item enters surveillance. Of interest is the estimation of the hazard function $\lambda(t) = -d\log S(t)/dt$, where $S(t) = \mathsf{P}(T > t)$ is the survival function.

For the accelerated life models, $\log T$ follows a location-scale family distribution, with

$$\lambda(t; \mu, \sigma) = \frac{1}{\sigma t}\frac{f(z)}{1 - F(z)}, \quad \text{for } z = \frac{\log t - \mu}{\sigma},$$

where $f(z)$ is a probability density on $(-\infty, \infty)$ and $F(z)$ is the associated distribution function; the Weibull family has $f(z) = we^{-w}$ for $w = e^z$, the log normal family has $f(z) = \phi(z)$, and the log logistic family has $f(z) = w/(1+w)^2$ for $w = e^z$. One is to estimate the location parameter $\eta = \mu$ as a function of the covariate u, and the minus log likelihood is given by

$$l(\eta; Y) = \delta \log\lambda(X; \mu, \sigma) - \int_Z^X \lambda(t; \mu, \sigma)dt,$$

where $Z = 0$ if there is no left-truncation.

Instead of a `Surv(...)` construct for `survreg`, the response for `gssanova` should be a matrix of two or three columns, with X_i in the first column, δ_i in the second, and Z_i in an optional third. The scale parameter σ is assumed to be common and is estimated along with $\eta(u)$, in the form of $\nu = \sigma^{-1}$.

The following sequence loads the Stanford heart transplant data included in **gss** and fits a Weibull regression, with the age at transplant as the covariate.

```
R> data("stan", package = "gss")
R> fit <- gssanova(cbind(time + 0.01, status) ~ age, data = stan,
+    family = "weibull")
```

One has a proportional hazard model in a Weibull fit,

$$\lambda(t,u) = \frac{\nu}{t}e^{z} = \nu t^{\nu-1}e^{-\nu\eta(u)} = \lambda_0(t)\lambda_1(u)$$

where the relative risk $\lambda_1(u)$ is defined up to a multiplicative constant. Cutting out $\eta_\emptyset$ from $\eta = \eta_\emptyset + \eta_u$, one may evaluate and plot the relative risk $e^{-\nu\eta_u(u)}$.

```
R> est <- predict(fit, stan, se = TRUE, inc = "age")
R> ix <- order(stan$age)
R> age0 <- stan$age[ix]
R> plot(age0, exp(-fit$nu * est$fit[ix]), type = "l", ylim = c(0,5))
R> lines(age0, exp(-fit$nu * (est$fit[ix] - 1.96 * est$se[ix])), col = 5)
R> lines(age0, exp(-fit$nu * (est$fit[ix] + 1.96 * est$se[ix])), col = 5)
```

`ssanova0` *and* `gssanova0`

The original `ssanova` and `gssanova` referred to in Gu (2002) have been renamed as `ssanova0` and `gssanova0`, which calculate the minimizer of (2) in $\mathcal{H}$ using the legacy **RKPACK** routines. The numerical treatments in **RKPACK** take advantage of a special structure that only holds for $q = n$, and the $O(n^3)$ algorithms in **RKPACK** often execute faster than the $O(nq^2)$ algorithms powering `ssanova`, `gssanova`, and `gssanova1`. Repeated calls to `ssanova0` or `gssanova0` yield identical results, as $q = n$ in this setting.

Unfortunately, important modeling tools such as the Kullback-Leibler projection and the mixed-effect models (see the section on optional arguments) are numerically incompatible with the algorithms implemented in **RKPACK**, and an $\alpha > 1$ in (5) is difficult to insert in the **RKPACK** routines. With limited capabilities compared to the alternatives, these legacy suites are largely obsolete.

`ssanova9`

For Gaussian regression with correlated data, $\mathbf{Y} \sim \mathcal{N}(\eta(\mathbf{x}), \sigma^2 W^{-1})$, where $\mathbf{Y} = (Y_1, \ldots, Y_n)^\top$ and $\eta(\mathbf{x}) = \big(\eta(x_1), \ldots, \eta(x_n)\big)^\top$, one may set

$$L(\eta) = \big(\mathbf{Y} - \eta(\mathbf{x})\big)^\top W \big(\mathbf{Y} - \eta(\mathbf{x})\big)$$

in (2), where W may depend on a few correlation parameters. One may use `ssanova9` in this setting, with the smoothing parameters selected along with correlation parameters by a cross-validation score derived in Han and Gu (2008).

An argument `cov` specifies W in `ssanova9` calls. For W^{-1} known, use `cov = list("known", w)`, where `w` contains W^{-1}. For longitudinal observations, use `cov = list("long", id)`, where `id` is a factor; when data at the same `id` levels are grouped together in $\mathbf{Y}$, $W^{-1} = I + \gamma ZZ^\top$, where $ZZ^\top$ is block-diagonal with blocks of the form $\mathbf{1}\mathbf{1}^\top$. For serially correlated data, one may set `cov = list("arma", c(p, q))` to use an ARMA(p, q) model for $\boldsymbol{\epsilon} = \mathbf{Y} - \eta(\mathbf{x})$.

More generally, one may enter W^{-1} via `cov = list(fun = fun, env = env, init = init)`, with W^{-1} to be evaluated by `cov$fun(gamma, cov$env)`; `env` contains constants and `init` provides initial values for the correlation parameters γ, which should be on scales free of constraint.

3.2. Density estimation suites

To estimate a probability density $f(x)$ on $\mathcal{X}$, one may use `ssden` or `ssden1`. To estimate a conditional density $f(y|x)$ on $\mathcal{X} \times \mathcal{Y}$, use `sscden` or `sscden1`. For $\mathcal{Y}$ discrete, one may use `ssllrm` to fit $f(y|x)$, which is regression with cross-classified responses.

`ssden`

A probability density $f(x)$ is positive and integrates to one. Consider a logistic density transform $f(x) = e^{\eta(x)}/\int_{\mathcal{X}} e^{\eta(x)}dx$, which can be made one-to-one by cutting out $\eta_\emptyset$ in a one-way ANOVA decomposition $\eta = \eta_\emptyset + \eta_x$. One may set

$$L(\eta) = -\frac{1}{n}\sum_{i=1}^{n} \eta(X_i) + \log \int_{\mathcal{X}} e^{\eta(x)}dx$$

in (2), where X_i are independent samples from $f(x)$.

The following sequence draws samples from a normal mixture and fits a cubic spline to the log density.

```
R> u <- runif(100)
R> x <- rnorm(100) * 0.1
R> x <- ifelse(u > 1/3, x + 0.7, x + 0.3)
R> fit <- ssden(~ x)
```

The domain $\mathcal{X}$ does contribute to estimation through $\int_{\mathcal{X}} e^{\eta(x)}dx$, and it is a good idea to specify $\mathcal{X}$ explicitly via `domain = data.frame(x = c(a, b))`; if unspecified, as is the case in the call above, the domain is set internally, extending the data range by 5% on both ends.

To use the fitted univariate density, one has the usual `d-`, `p-`, and `q-` functions, but no `r-`.

```
R> domain <- fit$domain$x
R> xx <- seq(domain[1], domain[2], length = 101)
R> plot(xx, dssden(fit, xx), type = "l")
R> qq <- qssden(fit, (0:10)/10)
R> pssden(fit, qq)
```

On product domains, ANOVA structures in log density may have conditional independence implications. A log density η on $\mathcal{X}_1 \times \mathcal{X}_2 \times \mathcal{X}_3$ containing all interactions can be fitted by

```
R> fit <- ssden(~ x1 * x2 * x3)
```

and one may check the Kullback-Leibler projection

```
R> project(fit, inc = c("x1", "x2", "x3", "x1:x3", "x2:x3"))
```

A log density of the form $\eta = \eta_1 + \eta_2 + \eta_3 + \eta_{13} + \eta_{23}$ implies conditional independence of $X_{\langle 1\rangle}$ and $X_{\langle 2\rangle}$ given $X_{\langle 3\rangle}$, or $(X_{\langle 1\rangle} \perp X_{\langle 2\rangle})|X_{\langle 3\rangle}$, to be fitted via

```
R> fit <- ssden(~ (x1 + x2) * x3)
```

The fitted density at `xx` can still be evaluated through `dssden(fit, xx)`, but `xx` here must be a data frame; `pssden` and `qssden` however are meaningless for a multivariate density. One may evaluate a "slice" of the fitted multivariate density via the implied conditional density, say $f(x_{\langle 1\rangle}|x_{\langle 2\rangle} = 0.5, x_{\langle 3\rangle} = 0.8)$, via

```
R> cond <- data.frame(x2 = 0.5, x3 = 0.8)
R> cdssden(fit, xx1, cond = cond)$pdf
R> qq <- cqssden(fit, (0:10)/10, cond = cond)
R> cpssden(fit, qq, cond = cond)
```

Additive models for the log density on product domains imply the mutual independence of the marginals, and in this circumstance, it is preferred to estimate the marginal densities separately.

`sscden`

Consider a logistic conditional density transform $f(y|x) = e^{\eta(x,y)}/\int_{\mathcal{Y}} e^{\eta(x,y)}dy$, which can be made one-to-one by cutting out $\eta_\emptyset + \eta_x$ in an ANOVA decomposition $\eta = \eta_\emptyset + \eta_x + \eta_y + \eta_{x,y}$. Observing (X_i, Y_i), $i = 1, \ldots, n$, one may set

$$L(\eta) = -\frac{1}{n}\sum_{i=1}^{n}\Big\{\eta(X_i, Y_i) - \log\int_{\mathcal{Y}} e^{\eta(X_i,y)}dy\Big\}$$

in (2) for the estimation of $f(y|x)$. Domains $\mathcal{X}$ and $\mathcal{Y}$ are generic, and can be product domains themselves, so η_y and $\eta_{x,y}$ may be further decomposed.

The following sequence loads the penny thickness data included in **gss** and fits a tensor product spline to the log conditional density.

```
R> data("penny", package = "gss")
R> fit <- sscden(~ year * mil, ~ mil, data = penny)
```

The first formula in a `sscden` call specifies model terms and the second formula lists the y-variables; $\eta_\emptyset$ and terms in η_x are removed internally. It would be a good idea to specify $\mathcal{Y}$ explicitly via `ydomain`.

The fitted $f(y|x)$ can be evaluated using `dsscden`, and when $\mathcal{Y}$ is a real interval, as is the case here, one may also use `psscden` and `qsscden`. The code below plots selected quantiles of $f(y|x)$ as functions of x.

```
R> quan <- qsscden(fit, c(0.05, 0.25, 0.5, 0.75, 0.95),
+    data.frame(year = penny$year))
R> plot(penny$year, penny$mil, col = 3)
R> for (i in 1:5) lines(penny$year, quan[i, ])
```

One may specify more than one y-variable in a `sscden` call, though at least one of them must be continuous. To evaluate conditional densities such as $f(y_{\langle 1\rangle}|x, y_{\langle 2\rangle})$, one has the `cdsscden`, `cpsscden`, and `cqsscden` functions.

`ssllrm`

For $\mathcal{Y}$ binary, the estimation of $f(y|x)$ reduces to logistic regression, and for $\mathcal{Y}$ discrete in general, it generalizes logistic regression to settings with cross-classified responses. To accommodate features specific to an all discrete $\mathcal{Y}$, `ssllrm` was created.
The following R function generates $Y|x$ on $\mathcal{Y} = \{0,1\}^2$, with $p_y(x) = \mathsf{P}(Y = y|x)$ given by

$$\begin{pmatrix} p_{0,0}(x) & p_{0,1}(x) \\ p_{1,0}(x) & p_{1,1}(x) \end{pmatrix} \propto \begin{pmatrix} 2q_1(x)q_2(x) & q_1(x)p_2(x) \\ p_1(x)q_2(x) & 2p_1(x)p_2(x) \end{pmatrix},$$

where $p_1(x) + q_1(x) = p_2(x) + q_2(x) = 1$; the odds ratio $p_{0,0}(x)p_{1,1}(x)/p_{1,0}(x)p_{0,1}(x) = 4$ is a constant here, independent of x.

```
R> rtest <- function(x) {
+    p1 <- plogis(400 * x^5 * (1 - x)^3 - 1)
+    p2 <- plogis(50 * x^2 * (1 - x)^4)
+    q1 <- 1 - p1; q2 <- 1 - p2
+    p <- cbind(2 * q1 * q2, q1 * p2, p1 * q2, 2 * p1 * p2)
+    y1 <- y2 <- NULL
+    for (i in 1:length(x)) {
+      ywk <- rmultinom(1, 1, p[i, ])
+      y1 <- c(y1, ywk[3] + ywk[4])
+      y2 <- c(y2, ywk[2] + ywk[4])
+    }
+    cbind(y1, y2)
+  }
```

The sequence below draws a sample and fits $f(y|x)$.

```
R> x <- runif(200)
R> ywk <- rtest(x)
R> y1 <- factor(ywk[, 1])
R> y2 <- factor(ywk[, 2])
R> fit <- ssllrm(~ x * y1 * y2, ~ y1 + y2)
```

where the y-variables must be factors. The log conditional density is of the form

$$\eta = \eta_y + \eta_{x,y} = \eta_1 + \eta_2 + \eta_{12} + \eta_{x1} + \eta_{x2} + \eta_{x12},$$

where $\eta_y(y_{\langle 1\rangle}, y_{\langle 2\rangle})$ is decomposed into $\eta_1 + \eta_2 + \eta_{12}$ and $\eta_{xy}(x, y_{\langle 1\rangle}, y_{\langle 2\rangle})$ into $\eta_{x1} + \eta_{x2} + \eta_{x12}$; the log odds ratio only involves $\eta_{12} + \eta_{x12}$. To check for $(Y_{\langle 1\rangle} \perp Y_{\langle 2\rangle})|X$, or $\eta_{12} + \eta_{x12} = 0$,

```
R> project(fit, inc = c("y1", "y2", "x:y1", "x:y2"))
```

and to check for a constant odds ratio, or $\eta_{x12} = 0$,

```
R> project(fit, inc = c("y1", "y2", "x:y1", "x:y2", "y1:y2"))
```

To evaluate the fitted $f(y|x)$ on a grid, use

```
R> xx <- seq(0, 1, length = 51)
R> predict(fit, data.frame(x = xx))
```

which returns a matrix with columns corresponding to y values as ordered in `fit$qd.pt`. Confidence intervals do not make sense for density estimates due to normalization, but for a y-contrast of $\log f(y|x)$,

$$\theta = \textstyle\sum_y c_y \log f(y|x) = \sum_y c_y \, \eta(x,y), \quad \sum_y c_y = 0,$$

the normalizing constant $\int_{\mathcal{Y}} e^{\eta(x,y)} dy$ cancels out. The log odds ratio is a y-contrast, and one may evaluate it on a grid along with standard errors

```
R> est <- predict(fit, data.frame(x = xx), odds = c(1, -1, -1, 1),
+    se = TRUE)
```

where `odds` specifies c_y in the order of `fit$qd.pt`.

`ssden1`

Integrals of the form $\int_{\mathcal{X}} h(x) e^{\eta(x)} dx$ have to be performed in `ssden`, which is numerically prohibitive for a high-dimensional $\mathcal{X}$. Jeon and Lin (2006) set

$$L(\eta) = \frac{1}{n} \sum_{i=1}^{n} e^{-\eta(X_i)} + \int_{\mathcal{X}} \eta(x) \rho(x) dx$$

in (2) for $\rho(x)$ a known density, with the resulting estimate $f(x) \propto e^{\eta(x)} \rho(x)$. When $\rho(x) = \prod_\gamma \rho_\gamma(x_{\langle\gamma\rangle})$ on $\mathcal{X} = \prod_\gamma \mathcal{X}_\gamma$, $\int_{\mathcal{X}} \eta(x)\rho(x)dx$ can be calculated as sums of products of univariate integrals, and ANOVA structures in $\eta(x)$ have the same conditional independence implications; the marginal density estimates on $\mathcal{X}_\gamma$ are natural choices for $\rho_\gamma(x_{\langle\gamma\rangle})$.

The approach is implemented in `ssden1` with the same user interface as `ssden`. Utility functions `dssden`, `cdssden`, `cpssden`, and `cqssden` also work with `ssden1` fits, except that the $f(x)$ values returned from `dssden` are unnormalized. A variant of Kullback-Leibler projection is implemented in `project`, one that avoids integrals of the form $\int_{\mathcal{X}} h(x) e^{\eta(x)} dx$.

`sscden1`

For the estimation of $f(y|x)$, integrals of the form $\int_{\mathcal{Y}} h(X_i, y) e^{\eta(X_i,y)} dy$ are performed repeatedly in `sscden`, which can be numerically formidable for n large despite the typically low dimension of $\mathcal{Y}$; the drag here is the number of distinctive X_i's. One may set

$$L(\eta) = \frac{1}{n} \sum_{i=1}^{n} \Big\{ e^{-\eta(X_i,Y_i)} + \int_{\mathcal{Y}} \eta(X_i,y) \rho(X_i,y) dy \Big\}$$

in (2) for some $\rho(x,y)$ satisfying $\int_{\mathcal{Y}} \rho(x,y)dy = 1$, $\forall x$, with the resulting estimate $f(y|x) \propto e^{\eta(x,y)}\rho(x,y)$. The term η_x is needed in η for the approach to work, though it cancels out in the fitted $f(y|x)$. The integrals $\int_{\mathcal{Y}} \eta(X_i,y)\rho(X_i,y)dy$ are linear combinations of integrals of basis functions, which only need to be computed once for all.

The approach is implemented in `sscden1`, with syntax the same as for `sscden` except for the specification of $\rho(x,y)$ via an argument `rho`. The default `rho = list("xy")` "estimates" $\rho(x,y)$ internally using `ssanova`, and `rho = list("y")` orders a marginal density estimate on $\mathcal{Y}$ from `ssden` for use as $\rho(x,y)$. One may also create $\rho(x,y)$ externally and pass it into `sscden1` through `rho = list(fun = fun, env = env)`, to be evaluated via `rho$fun(x, y, rho$env, outer)`, where `env` contains constants and `outer` indicates whether to return a vector of $\rho(x_i,y_i)$ or a matrix $\rho(\mathbf{x},\mathbf{y}^\top)$.

The utility functions for the evaluation of $f(y|x)$ also work for `sscden1` fits. A variant of Kullback-Leibler projection is implemented in `project`, but the tool is not as useful here; with $\rho(x,y)$ in the scene, conditional independence may not be inferable just from the ANOVA structures in $\eta(x,y)$.

3.3. Hazard estimation suites

Observing (X_i,δ_i,Z_i,U_i), $i=1,\ldots,n$, where $X=\min(T,C)$, $\delta=I_{[T\le C]}$, $Z<X$, and U is a possible covariate, one is to estimate the hazard $\lambda(t,u)=e^{\eta(t,u)}$. The accelerated life models via `gssanova` are parametric in t, and as more flexible alternatives, one may use `sshzd` or `sshzd1` for fully nonparametric estimation. In case a proportional hazard model holds, $\lambda(t,u)=\lambda_0(t)\lambda_1(u)$, one may also use `sscox` to estimate the relative risk $\lambda_1(u)$.

`sshzd`

With or without a covariate, one may set

$$L(\eta) = -\frac{1}{n}\sum_{i=1}^{n}\left\{\delta_i\eta(X_i,U_i) - \int_{Z_i}^{X_i} e^{\eta(t,U_i)}dt\right\}$$

in (2), where U_i drops out if absent in the data, and $Z_i=0$ if there is no left-truncation.

The following sequence fits a log hazard of the form $\eta=\eta_\emptyset+\eta_t+\eta_u+\eta_{t,u}$ to the Stanford heart transplant data.

```
R> data("stan", package = "gss")
R> fit <- sshzd(Surv(futime, status) ~ futime * age, data = stan)
```

Data scatter more evenly on the $t^*=\sqrt{t}$ scale, and `futime` is thus transformed; the hazard on the original time scale should be $e^{\eta(\sqrt{t},u)}/2\sqrt{t}$.

The `Surv(t, delta, z = 0)` construct has a similar appearance as that in the **survival** package, but is parsed differently here; the `t` main effect must appear among the model terms.

To assess the plausibility of a proportional hazard model, or an additive model in log hazard, check the Kullback-Leibler projection

```
R> project(fit, inc = c("futime", "age"))
```

To evaluate the fitted $e^{\eta(t^*,u)}$, say 400 days after transplant (so `futime = 20`) for a 30-year-old and 100 days after transplant for a 20-year-old, use

```
R> new <- data.frame(futime = c(20, 10), age = c(30, 20))
R> est <- hzdrate.sshzd(fit, new, se = TRUE)
```

where `est$fit` contains $e^{\hat{\eta}(t^*,u)}$ and `est$se` contains the standard error $s(t^*, u)$ of $\hat{\eta}(t^*, u)$. For hazard curves at given u values, say for a 20-year-old and a 30-year-old, try

```
R> tt <- seq(0, 60, leng = 51)
R> age0 <- c(20, 30)
R> hzdcurve.sshzd(fit, tt, data.frame(age = age0))
```

One may also calculate the expected survival via

```
R> survexp.sshzd(fit, tt, data.frame(age = age0))
```

which is simply the estimated survival probability $\hat{S}(t^*, u) = \exp\big\{ -\int_0^{t^*} e^{\hat{\eta}(s,u)}ds\big\}$.

`sscox`

For $\lambda(t, u) = \lambda_0(t)\lambda_1(u)$, one may treat the base hazard $\lambda_0(t)$ as a nuisance and estimate the relative risk $\lambda_1(u) = e^{\eta(u)}$, using in (2) the partial likelihood,

$$L(\eta) = -\frac{1}{n}\sum_{i=1}^{n}\delta_i\Big\{\eta(U_i) - \log\sum_{j=1}^{n} I_{[Z_j<X_i\le X_j]}e^{\eta(U_j)}\Big\}.$$

As $\lambda_1(u)$ is defined only up to a multiplicative constant, we choose to cut out $\eta_\emptyset$ from $\eta(u) = \eta_\emptyset + \eta_u$.

The following sequence estimates the relative risk of `age` in `stan` and evaluates the fit on the data points.

```
R> fit <- sscox(Surv(time, status) ~ age, data = stan)
R> risk <- predict(fit, data.frame(age = stan$age), se = TRUE)
```

Only the ordering of X_i matters here, so one may use either the original `time` or the transformed `futime` and get the same fit. The fitted relative risk $e^{\hat{\eta}_u(u_i)}$ are in `risk$fit` and the standard error $s(u_i)$ of $\hat{\eta}_u(u_i)$ are in `risk$se`. To estimate the base hazard given $r_i = \eta_u(u_i)$, one may use `sshzd` with offset.

```
R> fit.b <- sshzd(Surv(futime, status) ~ futime, data = stan,
+    offset = log(risk$fit))
```

which effectively uses in (2) the likelihood

$$L(\eta) = -\frac{1}{n}\sum_{i=1}^{n}\Big\{\delta_i\eta(X_i) - \int_{Z_i}^{X_i} e^{\eta(t)+r_i}dt\Big\}.$$

The base hazard is easily evaluated on a grid.

```
R> hzd.b <- hzdcurve.sshzd(fit.b, tt, se = TRUE)
```

Note that the relative risk and the base hazard in a proportional hazard model can be estimated jointly using `sshzd` through penalized full likelihood.

```
R> fit0 <- sshzd(Surv(futime, status) ~ futime + age, data = stan)
R> risk0 <- hzdrate.sshzd(fit0, data.frame(age = stan$age), se = TRUE,
+    inc = "age")
R> hzd.b0 <- hzdrate.sshzd(fit0, data.frame(futime = tt), se = TRUE,
+    inc = c("1", "futime"))
```

`sshzd1`

With continuous covariates U_i, integrals of the form $\int_{Z_i}^{X_i} h(t,U_i)e^{\eta(t,U_i)}dt$ are performed repeatedly in `sshzd`, putting it under similar numerical burdens as `sscden`. As an alternative, one may use

$$L(\eta) = \frac{1}{n}\sum_{i=1}^{n}\Big\{\delta_i e^{-\eta(X_i,U_i)}\rho(X_i,U_i) + \int_{Z_i}^{X_i}\eta(t,U_i)\rho(t,U_i)dt\Big\}$$

in (2); integrals $\int_{Z_i}^{X_i}\eta(t,U_i)\rho(t,U_i)dt$ are linear combinations of integrals of basis functions, which only need to be computed once for all. The hazard estimate is still $e^{\eta(t,u)}$, not too sensitive to the choice of $\rho(t,u)$, which essentially serves as weights in estimation.

The approach is implemented in `sshzd1`, with the same syntax and utility functions as `sshzd`. The default `rho = list("marginal")` uses as $\rho(t,u)$ a covariate-free hazard estimate from `sshzd`, and `rho = list("weibull")` fits a Weibull regression using `gssanova` and uses as $\rho(t,u)$ the resulting hazard estimate $\lambda(t,u) = \nu t^{\nu-1}e^{-\nu\eta(u)}$.

4. Model configurations

Model terms are specified by the model formula as in `lm`, and schematically, the configurations of the terms are through the construction of the "inverse" of $J(\eta) = \sum_\beta \theta_\beta^{-1}J_\beta(\eta_\beta)$, $R_J(x,y) = \sum_\beta \theta_\beta R_\beta(x,y)$. For η_β a main effect, say $\eta_1(x_{\langle 1\rangle})$, $R_\beta(x,y) = R_{\langle 1\rangle}(x_{\langle 1\rangle}, y_{\langle 1\rangle})$ is simply a kernel on $\mathcal{X}_1$, and for η_β an interaction, say $\eta_{12}(x_{\langle 1\rangle}, x_{\langle 2\rangle})$, $R_\beta(x,y) = R_{\langle 1\rangle}(x_{\langle 1\rangle}, y_{\langle 1\rangle})R_{\langle 2\rangle}(x_{\langle 2\rangle}, y_{\langle 2\rangle})$ is the product of kernels on $\mathcal{X}_1$ and $\mathcal{X}_2$. Besides the model formula, model configurations are through the specifications of the marginal kernels.

4.1. Numerical vectors

For `x` a numerical vector $x_{\langle\gamma\rangle} \in \mathcal{X}_\gamma = [a,b]$, the default marginal kernel is the "inverse" of $\int_a^b \big(f''(x)\big)^2 dx$, and terms η_β involving $x_{\langle\gamma\rangle}$ satisfy side conditions $\int_a^b \eta_\beta(x)dx_{\langle\gamma\rangle} = 0$. The domain may be set internally by extending the data range 5% on each end, or may be specified explicitly via

```
  type = list(x = list("cubic", c(a, b)))
```

In density estimation suites, specifications via `domain` or `ydomain` also get used in kernel generation.

A useful alternative to the default kernel is a periodic cubic spline kernel, the "inverse" of $\int_a^b \big(f''(x)\big)^2 dx$ for $f(x)$ periodic on $[a, b]$. Such a kernel is specified via

```
  type = list(x = list("per", c(a, b)))
```

and terms involving such a $x_{\langle\gamma\rangle}$ are periodic in $x_{\langle\gamma\rangle}$.

4.2. Numerical matrices

The marginal domains $\mathcal{X}_\gamma$ are generic, which can be mathematically multi-dimensional. Two such cases are the Euclidean space $(-\infty, \infty)^d$ and the unit sphere $\mathcal{S}$. With a matrix `x` in the model formula, one may entertain multi-dimensional marginals, usually geographic/spatial, in tensor product splines.

The default kernel for a numerical matrix `x` is a thin-plate spline kernel of order $m = 2$, which, for $x_{\langle\gamma\rangle} = z = (z_{\langle 1\rangle}, z_{\langle 2\rangle}) \in (-\infty, \infty)^2$, is the "inverse" of

$$\int\int \Big(\frac{\partial^2 f}{\partial z_{\langle 1\rangle}^2} + 2\frac{\partial^2 f}{\partial z_{\langle 1\rangle}\partial z_{\langle 2\rangle}} + \frac{\partial^2 f}{\partial z_{\langle 2\rangle}^2}\Big)^2 dz_{\langle 1\rangle} dz_{\langle 2\rangle}.$$

One needs $2m > d$, and to specify an order $m \neq 2$, use

```
  type = list(x = list("tp", m))
```

Unless more detailed customization is done via `type`, terms η_β involving such an $x_{\langle\gamma\rangle}$ add up to zero on the data points along $x_{\langle\gamma\rangle}$, $\sum_i \eta_\beta(\dots, x_{i\langle\gamma\rangle}, \dots) = 0$. Thin-plate splines are invariant to arbitrary shifting and rotation of the coordinate system on the domain.

For `x` with two columns, one may alternatively treat it as $x_{\langle\gamma\rangle} \in \mathcal{S}$ and specify a spherical spline via

```
  type = list(x = list("sphere", m))
```

where $m = 2, 3, 4$ with 2 the default order; technical details are to be found in Wahba (1981). The first column in `x` should be the latitude in the range $[-90, 90]$ and the second column the longitude in the range $[-180, 180]$. Spherical splines are invariant to rotations of the coordinates on $\mathcal{S}$, and terms η_β involving such an $x_{\langle\gamma\rangle}$ satisfy $\int_{\mathcal{S}} \eta_\beta(x) dx_{\langle\gamma\rangle} = 0$.

To preserve a matrix element in a data frame, one may use the as-is function `I(...)`, as shown below.

```
R> test <- data.frame(x = 1:5, y = I(cbind(1:5, 5:9)))
R> test[, 2]
R> test$y
```

4.3. Factors

On a discrete $\mathcal{X}_\gamma$, a function $f(x)$ is a vector and a reproducing kernel $R(x, y)$ is a square matrix, and a quadratic "roughness" functional can be written as a quadratic form $J(f) =$

$f^\top J f$ for J a nonnegative-definite matrix. In the column space of J with the square norm $f^\top J f$, the reproducing kernel is J^+, the Moore-Penrose inverse of J.

Discrete variables enter the model formula as factors. For `x` nominal, the kernel is the one associated with $J(f) = \sum_x \big(f(x) - \bar{f}\big)^2$, where $\bar{f} = \sum_x f(x) / \sum_x 1$. For `x` ordinal on $\{1, \ldots, K\}$, the kernel is associated with $J(f) = \sum_{x=2}^K \big(f(x) - f(x-1)\big)^2$. For `x` binary, there is no distinction.

5. Optional arguments

We now discuss a couple of optional arguments that help to broaden the scope of application.

5.1. Parametric terms

In some applications, semiparametric models of the form $\zeta(x, \mathbf{z}) = \eta(x) + \mathbf{z}^\top \boldsymbol{\beta}$ are desirable, where η is "nonparametric" and $\mathbf{z}^\top \boldsymbol{\beta}$ comprises parametric terms. Replacing η by ζ in the likelihood $L(\eta)$ in (2), and minimizing the resulting functional with respect to η and $\boldsymbol{\beta}$, one obtains the so-called partial splines.

Partial spline models can be fitted in the regression and hazard estimation suites; due to normalization, such models make little sense for density estimation. To enter parametric terms $z_1\beta_1 + z_2\beta_2 + z_1 z_2 \beta_{12}$, say, use

```
partial = ~ z1 * z2
```

which acts just like a formula in `lm`. Variables in `partial` are expected to be numerical vectors, and the partial terms are centralized internally; centralization serves the same purpose as the averaging operators for the ANOVA terms. Like the ANOVA terms, the partial terms may also be selectively included/excluded in `predict` or `hzdrate.sshzd`. To maintain model identifiability, one should avoid "overlaps" among the variables in x and $\mathbf{z}$.

5.2. Random effects

Mixed-effect models are widely used for the modeling of correlated data, such as those arising in longitudinal studies. Consider a mixed-effect model $\zeta(x, \mathbf{z}) = \eta(x) + \mathbf{z}^\top \mathbf{b}$, where $\mathbf{b} \sim \mathcal{N}(\mathbf{0}, B)$. One may estimate η and $\mathbf{b}$ jointly via the minimization of

$$L(\zeta) + \lambda\, J(\eta) + \mathbf{b}^\top \Sigma \mathbf{b},$$

where $\Sigma \propto B^{-1}$. Mixed-effect models for the log hazard are known as frailty models.

The random effects $\mathbf{z}^\top \mathbf{b}$ in a mixed-effect model appear identical to the parametric terms $\mathbf{z}^\top \boldsymbol{\beta}$ in a partial spline model, except that they need to be penalized via $\mathbf{b}^\top \Sigma \mathbf{b}$. The matrix Σ is typically structured containing correlation parameters γ, which are to be selected along with the smoothing parameters.

The term $\mathbf{z}^\top \boldsymbol{\beta}$ in partial spline models are often more important than $\eta(x)$, but the random effects $\mathbf{z}^\top \mathbf{b}$ are a nuisance and are usually set to zero when the fits are evaluated. These terms may coexist, but one would need different symbols to distinguish the two $\mathbf{z}$'s.

Random effects are specified via $Z = (\mathbf{z}_1, \ldots, \mathbf{z}_n)^\top$ and Σ. One may use an optional argument `random` in the regression and hazard estimation suites, as a formula or a list. With

```
random = ~ 1 | id
```

where `id` is a factor with p levels, $Z = \mathrm{diag}(\mathbf{1}, \ldots, \mathbf{1})$ when data at the same `id` levels are grouped together, with the $\mathbf{1}$'s possibly of different lengths, and $\Sigma = \gamma I_p$ with a single γ; this typically characterizes correlations in longitudinal data. With

```
random = ~ gid | id
```

where `gid` is either the same as `id` or a coarser grouping with levels collapsed from those of `id`, Z is the same as above, but Σ is now block-diagonal with blocks of the form $\gamma_k I_{p_k}$, for $\sum_k p_k = p$; this can be used to characterize correlations in clustered data. More generally, one may use a list

```
random = list(z = z, sigma = list(fun, env), init = init)
```

where `z` contains Z, `sigma` specifies Σ to be evaluated via

```
sigma$fun(gamma, sigma$env)
```

and `init` holds initial values for γ, presumably on scales free of constraint.

For Gaussian regression with longitudinal data, `ssanova9` with `cov = list("long", id)` should be used when $p \asymp n$, and `ssanova` with `random = ~ 1 | id` should be used when $p = O(\sqrt{n})$.

Random effects find little use in density estimation due to normalization, but $\mathbf{z}^\top \mathbf{b}$ for a univariate ζ can be propagated into multivariate versions for use in `ssllrm` (Gu and Ma 2011), which accepts `random`.

6. Numerical engines

Apart from `ssanova0`, `gssanova1`, and `gssanova0`, the estimation is done in two nested loops, with the outer loop minimizing a cross-validation score $V(\lambda)$ with respect to smoothing parameters plus possible correlation parameters, and with the inner loop minimizing (2) with fixed tuning parameters. The inner loop follows Newton iteration with safeguards such as step-halving, and the outer loop uses the quasi-Newton iteration of Dennis and Schnabel (1996) as implemented in the R function `nlm`; the inner loop is absent in `ssanova` where the estimate is analytical.

It can be shown that for $\eta \in \mathcal{H}^*$ as in (4), $J(\eta) = \mathbf{c}^\top R_J(\mathbf{z}, \mathbf{z}^\top)\mathbf{c} = \mathbf{c}^\top \big(\sum_\beta \theta_\beta Q_\beta\big)\mathbf{c}$, where $Q_\beta = R_\beta(\mathbf{z}, \mathbf{z}^\top)$ and c_j are coefficients of the basis functions $R_J(z_j, \cdot)$. Fixing the θ's in $J(\eta) = \sum_\beta \theta_\beta^{-1} J_\beta(\eta_\beta)$, the outer loop with a single λ is a simple task, and the starting values for the θ iteration via quasi-Newton are obtained through two passes of fixed-θ outer loop:

1. Set $\breve{\theta}_\beta^{-1} \propto \mathrm{tr}(Q_\beta)$, then minimize $V(\lambda)$ with respect to λ to obtain $\breve{\eta}$.

2. Set $\tilde{\theta}_\beta \propto J_\beta(\breve{\eta}_\beta)$, then minimize $V(\lambda)$ with respect to λ to obtain $\tilde{\eta}$.

Step 1 is invariant to arbitrary scalings of $J_\beta(\eta_\beta)$, allowing equal opportunity for all. Step 2 grants more allowances to terms that showed strengths in Step 1. The ensuing θ iteration fixes λ and starts from $\tilde{\theta}_\beta$.

The starting values $\tilde{\theta}_\beta$ proved to be highly effective, often leaving only the "last 20%" performance for the θ iteration to pick up. When the number of θ_β's are large, quasi-Newton iteration with numerical derivatives is slow to converge, and one may simply take $\tilde{\eta}$ and forgo the θ iteration. This can be done by setting `skip = TRUE` in the fitting functions.

If correlation parameters are involved in the process, as is the case when `random` is specified or when `ssanova9` is called upon, the computational savings via `skip = TRUE` would be less dramatic.

7. Examples

Three examples follow, one involving longitudinal data, another concerning density estimation on a truncated domain, and a third having a two-dimensional marginal domain.

7.1. Treatment of bacteriuria

A group of 72 patients with acute spinal cord injury and bacteriuria (bacteria in urine) were randomly assigned to two treatment groups, 36 each. A weekly binary indicator of bacteriuria was recorded for every patient over 4 to 16 weeks. The data are listed in Joe (1997, Section 11.4). Removing the week-1 data, in which the disease indicator is positive for all, one has $n = 820$. The data are included in **gss** as a data frame.

```
R> data("bacteriuria", package = "gss")
R> bact <- bacteriuria
```

There are 30 distinctive x_i's (15 `time` points by 2 `trt` levels), and `id` 3 and 38 had complete follow-up under the two `trt` levels, so the selection of $\{z_j\}$ can be done deterministically.

```
R> id.z <- (1:820)[bact$id %in% c(3, 38)]
```

An initial logistic regression model is fitted by

```
R> fit0 <- gssanova(infect ~ trt * time, "binomial", data = bact,
+    random = ~ 1 | id, id.basis = id.z)
```

with the logit in $\zeta(x, s) = \eta(x) + b_s$, where s is `id` levels and b_s are subject random effects. The interaction is negligible, so an additive model is fitted.

```
R> project(fit0, c("trt", "time"))
R> fit1 <- gssanova(infect ~ trt + time, "binomial", data = bact,
+    random = ~ 1 | id, id.basis = id.z)
```

`id` 1–36 were under `trt` 1 and `id` 37–72 were under `trt` 2, and a quick check on the subject random effects reveals disparity between the two `trt` levels.

```
R> var(fit1$b[1:36])

[1] 0.05118155
```

```
R> var(fit1$b[37:72])

[1] 0.2275906
```

`trt` 1 seems to allow less "individualism," so it appears appropriate to attach separate γ's.

```
R> fit2 <- gssanova(infect ~ trt + time, "binomial", data = bact,
+    random = ~ trt | id, id.basis = id.z)
R> var(fit2$b[1:36])

[1] 1.582536e-15
```

The subject effects are effectively absent under `trt` 1. The estimated infection probability as a function of `time`, under `trt` 1, say, can be evaluated and plotted.

```
R> new <- data.frame(trt = factor(rep(1, 15)), time = 2:16)
R> est1 <- predict(fit2, new, se = TRUE)
R> plot(2:16, plogis(est1$fit), type = "l", ylim = c(0, 1))
R> lines(2:16, plogis(est1$fit - 1.96 * est1$se), col = 5)
R> lines(2:16, plogis(est1$fit + 1.96 * est1$se), col = 5)
```

7.2. AIDS incubation

To study AIDS incubation, a valuable source of information is in the records of patients who were infected with HIV through blood transfusion, of which the date can be ascertained retrospectively. A data set is listed in Wang (1989), which includes the time X from transfusion to the diagnosis of AIDS, the time Y from transfusion to the end of study, and the age of the individual at transfusion; it is clear that $X \le Y$. The data are included in **gss** as a data frame, and we use the subset with `age` at 60 or above.

```
R> data("aids", package = "gss")
R> aids1 <- aids[aids$age >= 60, ]
```

Conditioning on the truncation mechanism, the density is $f(x,y) = e^{\eta(x,y)}/\int_{\mathcal{T}} e^{\eta(x,y)}dxdy$, where $\mathcal{T} = \{x < y\}$. The domain $\mathcal{T}$ enters estimation through $\int_{\mathcal{T}} e^{\eta(x,y)}dxdy$, effectively specified via the quadrature. Lacking better alternatives, we use a regular grid on $[0,100]^2$, cutting out points on $\{x > y\}$, and halving weights along $\{x = y\}$.

```
R> qd.pt <- expand.grid(incu = 2 * (1:50) - 1, infe = 2 * (1:50) - 1)
R> qd.pt <- qd.pt[qd.pt$incu <= qd.pt$infe, ]
R> qd.wt <- rep(1, nrow(qd.pt))
R> qd.wt[qd.pt$incu == qd.pt$infe] <- 0.5
R> qd.wt <- qd.wt / sum(qd.wt) * 5e3
R> quad <- list(pt = qd.pt, wt = qd.wt)
```

To fit a log density of the form $\eta = \eta_x + \eta_y + \eta_{x,y}$, use

```
R> domain <- data.frame(incu = c(0, 100), infe = c(0, 100))
R> fit0 <- ssden(~ incu * infe, data = aids1, domain = domain, quad = quad)
```

where quad overrides the internal generation of quadrature for a rectangular domain. Checking

```
R> project(fit0, c("incu", "infe"))
```

pre-truncation independence appears plausible, so an additive model can be fitted.

```
R> fit1 <- ssden(~ incu + infe, data = aids1, domain = domain, quad = quad)
```

Note that an additive model should never be fitted to a log density on a rectangular domain; independent marginal densities are best estimated separately in univariate settings.
The fitted incubation density $f(x)$ and infection density $f(y)$ are easily evaluated on a grid.

```
R> xx <- 2 * (1:50) - 1
R> cdssden(fit1, xx, cond = data.frame(infe = 50))$pdf
R> cdssden(fit1, xx, cond = data.frame(incu = 50))$pdf
```

7.3. Water acidity in lakes

From the Eastern Lake Survey of 1984 conducted by EPA, Douglas and Delampady (1990) derived a data set containing geographic locations, water acidity levels, and main ion concentrations in 1798 lakes. A subset of the data concerning 112 lakes in the Blue Ridge is included in **gss** as a data frame.

```
R> data("LakeAcidity", package = "gss")
R> acid <- LakeAcidity
```

A model of the form $\eta = \eta_\emptyset + \eta_c + \eta_g + \eta_{c,g}$ is fitted

```
R> fit0 <- ssanova(ph ~ log(cal) * geog, data = acid)
```

where geog, a matrix with two columns, contains the x-y coordinates of the lakes with respect to a local origin, converted from longitude-latitude. The term $\eta_{c,g}$ appears negligible, so an additive model is fitted.

```
R> project(fit0, c("log(cal)", "geog"))
R> fit1 <- ssanova(ph ~ log(cal) + geog, data = acid)
```

The geog main effect can be evaluated on a grid

```
R> xx0 <- seq(-0.04, 0.04, len = 31)
R> xx <- cbind(rep(xx0, 31), rep(xx0, rep(31, 31)))
R> est.g <- predict(fit1, data.frame(geog = I(xx)), se = TRUE, inc = "geog")
```

and in turn be plotted as contours

```
R> ff <- matrix(est.g$fit, 31, 31)
R> filled.contour(xx0, xx0, ff, asp = 1, plot.axes = points(acid$geog))
R> se <- matrix(est.g$se, 31, 31)
R> filled.contour(xx0, xx0, se, asp = 1, plot.axes = points(acid$geog)})
```

8. Miscellaneous

Optional arguments affiliated with model formulas, such as `subset` and `weights`, are accepted by the fitting functions in **gss**, where `weights` specifies multiplicity counts for duplicated data, except in Gaussian regression where it calls for weighted least squares. The regression suites also accept `offset`.

The cosine diagnostics of Gu (1992b) can be obtained for fitted regression models by using the method `summary`, with a flag `diag = TRUE`.

For density estimation using data that are subject to sampling bias, one may use `ssden` with an optional argument `bias`. Further details can be found in Gu (2013, Sections 7.6.4–7.6.5).

In case one needs to use marginal kernels that are not "canned" in **gss**, "custom" types can be defined for variables and be entered via `type = list(x = list("custom", ...))`. Further details can be found in Gu (2013, Section A.1.3).

9. Summary

The **gss** package offers nonparametric modeling tools in many of the standard data analysis settings. It also defines some non-standard settings, such as regression with cross-classified responses.

Functional ANOVA decompositions are built-in on multivariate domains, with continuous, discrete, and even multi-dimensional marginals. Semiparametric models and mixed-effect models can also be entertained. Smoothing parameters are selected by cross-validation, and the "testing" of model terms can be done using Kullback-Leibler projection.

References

Craven P, Wahba G (1979). "Smoothing Noisy Data with Spline Functions: Estimating the Correct Degree of Smoothing by the Method of Generalized Cross-Validation." *Numerische Mathematik*, **31**(4), 377–403.

Dennis JE, Schnabel RB (1996). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, Philadelphia. Corrected reprint of the 1983 original.

Douglas A, Delampady M (1990). "Eastern Lake Survey – Phase I: Documentation for the Data Base and the Derived Data Sets." *Technical Report 160 (SIMS)*, Department of Statistics, University of British Columbia, Vancouver, BC.

Gu C (1989). "**RKPACK** and Its Applications: Fitting Smoothing Spline Models." In *ASA Proceedings of the Statistical Computing Section*, pp. 42–51.

Gu C (1992a). "Cross-Validating Non-Gaussian Data." *Journal of Computational and Graphical Statistics*, **1**(2), 169–179.

Gu C (1992b). "Diagnostics for Nonparametric Regression Models with Additive Term." *Journal of the American Statistical Association*, **87**(420), 1051–1058.

Gu C (2002). *Smoothing Spline ANOVA Models*. Springer-Verlag, New York.

Gu C (2004). "Model Diagnostics for Smoothing Spline ANOVA Models." *Canadian Journal of Statistics*, **32**(4), 347–358.

Gu C (2013). *Smoothing Spline ANOVA Models*. 2nd edition. Springer-Verlag, New York.

Gu C, Ma P (2011). "Nonparametric Regression with Cross-Classified Responses." *Canadian Journal of Statistics*, **39**(4), 591–609.

Gu C, Wahba G (1993). "Smoothing Spline ANOVA with Component-Wise Bayesian "Confidence Intervals"." *Journal of Computational and Graphical Statistics*, **2**(1), 97–117.

Gu C, Wang J (2003). "Penalized Likelihood Density Estimation: Direct Cross-Validation and Scalable Approximation." *Statistica Sinica*, **13**(3), 811–826.

Han C, Gu C (2008). "Optimal Smoothing with Correlated Data." *Sankhya A*, **70**(1), 38–72.

Hastie TJ, Tibshirani RJ (1990). *Generalized Additive Models*. Chapman & Hall, London.

Jeon Y, Lin Y (2006). "An Effective Method for High Dimensional Log-Density ANOVA Estimation, with Application to Nonparametric Graphical Model Building." *Statistica Sinica*, **16**(2), 353–374.

Joe H (1997). *Multivariate Models and Dependence Concepts*. Chapman & Hall, London.

Kim YJ, Gu C (2004). "Smoothing Spline Gaussian Regression: More Scalable Computation via Efficient Approximation." *Journal of the Royal Statistical Society B*, **66**(2), 337–356.

Kimeldorf G, Wahba G (1971). "Some Results on Tchebycheffian Spline Functions." *Journal of Mathematical Analysis and Applications*, **33**(1), 82–85.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Therneau TM (2014). *A Package for Survival Analysis in S*. R package version 2.37-7, URL http://CRAN.R-project.org/package=survival.

Therneau TM, Grambsch PM (2000). *Modeling Survival Data: Extending the Cox Model*. Springer-Verlag, New York.

Wahba G (1981). "Spline Interpolation and Smoothing on the Sphere." *SIAM Journal on Scientific and Statistical Computing*, **2**(1), 5–16.

Wahba G (1983). "Bayesian "Confidence Intervals" for the Cross-Validated Smoothing Spline." *Journal of the Royal Statistical Society B*, **45**(1), 133–150.

Wahba G (1985). “A Comparison of GCV and GML for Choosing the Smoothing Parameter in the Generalized Spline Smoothing Problem.” *The Annals of Statistics*, **13**(4), 1378–1402.

Wang MC (1989). “A Semiparametric Model for Randomly Truncated Data.” *Journal of the American Statistical Association*, **84**(407), 742–748.

Affiliation:

Chong Gu
Department of Statistics
Purdue University
West Lafayette, IN 47907, United States of America
E-mail: chong@stat.purdue.edu

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 58, Issue 5 *Submitted:* 2012-04-02
June 2014 *Accepted:* 2014-02-09