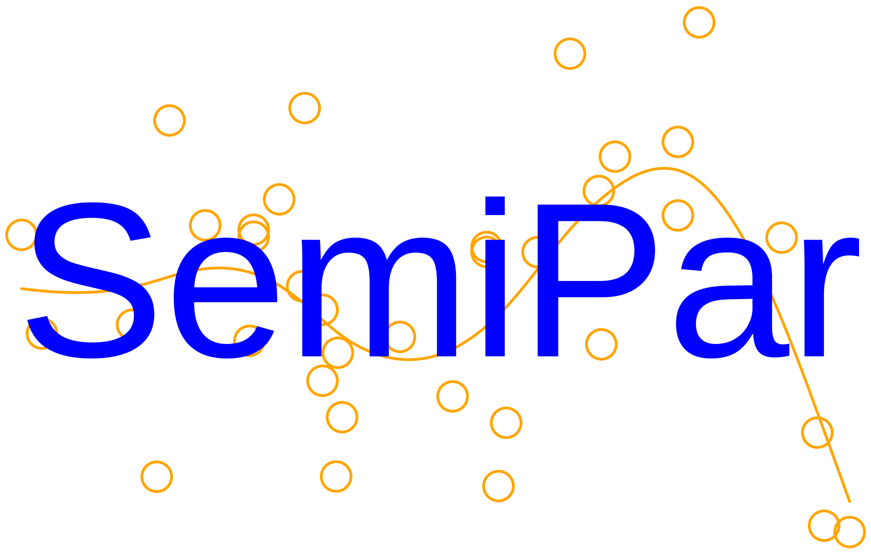


SemiPar

An R Package for Semiparametric Regression

Version 1.0

1st February, 2009

This document is best viewed in COLOUR

About SemiPar

SemiPar is an R language package for aiding semiparametric regression analyses, and accompanies the book:

Ruppert, D., Wand, M. P. and Carroll, R.J. (2003). *Semiparametric Regression*. New York: Cambridge University Press.

The current version of **SemiPar** may be downloaded from the Comprehensive R Archive Network (`http://cran.r-project.org`).

The **SemiPar** project is headed by

M. P. Wand,
School of Mathematics and Applied Statistics,
University of Wollongong,
Wollongong, Australia

to whom correspondence should be addressed (`wand@maths.unsw.edu.au`).

Other contributors to the project are

B.A. Coull,
Department of Biostatistics,
Harvard School of Public Health,
Boston, USA

J.L. French,
Pfizer, Inc.
Groton, Connecticut,
USA

B. Ganguli,
Department of Statistics,
University of Calcutta,
Kolkata, India

E.E. Lake,
Eigenstat, Inc.
Newton, Massachusetts,
USA

J. Staudenmayer,
Department of Mathematics,
University of Massachusetts,
Amherst, USA

A. Zanobetti,
Department of Environmental Health,
Harvard School of Public Health,
Boston, USA

The groundwork for **SemiPar** was laid through affiliations of all contributors with the Department of Biostatistics, Harvard School of Public Health and its

support is gratefully acknowledged. Support was provided by U.S. National Institute of Environmental Health Sciences grants R01-ES10844-01 and T32-ES07142-19.

Citing SemiPar

The following citation is suggested for projects that make use of **`SemiPar`** 1.0:

Wand, M.P., Coull, B.A., French, J.L., Ganguli, B., Kammann, E.E., Staudenmayer, J. and Zanobetti, A. (2005). SemiPar 1.0. R package. `http://cran.r-project.org`

Acknowledgements

We are grateful to users of earlier versions of **`SemiPar`** who contributed comments and suggestions. Particular special thanks goes to Rob Hyndman and Chris Paciorek.

About Version 1.0

Version 1.0 is the first public version of the **`SemiPar`** package and contains the more developed ("time tested") components. Since the release of **`SemiPar`** in 2005 the project head has had difficulty, time-wise, fixing up some (mainly minor) glitches that have been found. In early 2009 it is not clear at all if, and when, these will be fixed. Information about these glitches is posted, sporadically, on the **`SemiPar`** web-site (see below). Please send e-mail to `mwand@uow.edu.au`.

The **`SemiPar`** web-site

`www.uow.edu.au/~mwand/SemiPar.html`

contains information on the current status of **`SemiPar`**.

SemiPar 1.0 Users' Manual

By B. Ganguli and M.P. Wand

1 Principal Functions

The principal functions in **SemiPar** are

`spm()`	obtain semiparametric model fit
`summary()`	summarise the fit numerically
`plot()`	display the fit graphically
`predict()`	obtain fitted values (and standard errors) over a specified region of the predictor space

The function that most resembles `spm()` is `gam()`, for fitting generalised additive models. The `gam()` function has been available in the `S-PLUS` language for many years and has recently become available in R via the package `gam` maintained by Trevor Hastie.

When the **SemiPar** project started in the late 1990s the main motivation was to add some features not possessed by `gam()`. Table 1 summarises the advantages and disadvantages of `spm()` when compared with `gam()`.

Table 1: Summary of comparison between `spm()` and `gam()`.

Advantages of `spm()` over `gam()`
Degrees of freedom may be estimated from data via REML
Likelihood ratio tests are more readily performed.
A bivariate term may be included in the model.
REML-based random intercepts may included in the model.
An `spm()` fit object involving a bivariate term can be plotted using `plot()`.
Derivative plots are supported by `plot()`.

Advantages of `gam()` over `spm()`
Time-tested, with most glitches ironed out.
Current `spm()` does not support overdispersion.
Faster, although `spm()` speed is quite reasonable.
Currently there is no `data` argument in `spm()`.

In more recent years the R package `mgcv` has emerged; maintained by Simon N. Wood. It also has a function named `gam()` which does have some of the features not possessed by the original `gam()`. For example, it does allow for freedom to be estimated from data via generalised cross-validation (GCV).

Another R package `lmeSplines` has similarities with **SemiPar** in that it takes advantage of the mixed model representation of spline-based smoothers. The R package `fields` for spatial data analysis also has some common ground with **`SemiPar`**.

2 Single Predictor Models

Consider the univariate *scatterplot smoothing* (or *nonparametric regression*) setting

$$y_i = f(x_i) + \varepsilon_i$$

where the (x_i, y_i), $1 \leqslant i \leqslant n$, are the scatterplot data, ε_i are zero mean random variables with variance σ_ε^2 and $f(x) = \mathsf{E}(y|x)$ is a smooth function.

In **`SemiPar`** f is estimated using penalised spline smoothing. Penalised spline smoothers come in a number of forms (e.g. Eilers and Marx, 1996; Ruppert and Carroll, 2000). The `spm()` default are based on radial basis functions, and may be viewed as a generalisation of smoothing splines (French, Kammann and Wand, 2001). The underlying model for $f(x)$ is the mixed model

$$f(x) = \sum_{j=0}^{m-1} \beta_j x^j + \sum_{k=1}^{K} u_k |x - \kappa_k|^{2m-1}, \quad m = 1, 2, 3, \ldots \tag{1}$$

with

$$\mathbf{u} \equiv [u_1, \ldots, u_K]^T \sim N(\mathbf{0}, \sigma_u^2 \mathbf{\Omega}^{-1/2} (\mathbf{\Omega}^{-1/2})^T), \quad \mathbf{\Omega} \equiv [|\kappa_k - \kappa_{k'}|^{2m-1}]_{1 \leqslant k, k' \leqslant K}. \tag{2}$$

The mixed model representation of penalised spline smoothers allows for automatic fitting using the `R` linear mixed model function `lme()`. Smoothing parameter selection can be done via restricted maximum likelihood (REML) and $\widehat{f}(x)$ can be obtained via estimated best linear unbiased prediction (EBLUP) (e.g. Robinson, 1991).

This class of penalised spline smoothers may also be expressed as

$$\widehat{\mathbf{f}} = \mathbf{C}(\mathbf{C}^T\mathbf{C} + \lambda^{2m-1}\mathbf{D})^{-1}\mathbf{C}^T\mathbf{y} \tag{3}$$

where $\lambda \equiv \sigma_u^2 / \sigma_\varepsilon^2$ is a so-called *smoothing parameter*,

$$\mathbf{C} \equiv [1 \; x_i \ldots x_i^{m-1} \; |x_i - \kappa_k|^{2m-1}]_{\substack{1 \leqslant i \leqslant n, \\ 1 \leqslant k \leqslant K}} \quad \text{and} \quad \mathbf{D} \equiv \begin{bmatrix} \mathbf{0}_{2\times 2} & \mathbf{0}_{2\times K} \\ \mathbf{0}_{K\times 2} & (\mathbf{\Omega}^{1/2})^T \mathbf{\Omega}^{1/2} \end{bmatrix}.$$

2.1 Automatic scatterplot smoothing

We shall first consider automatic scatterplot smoothing of the fossil data collected by Bralower *et al.*(1997) and analysed by Chaudhuri and Marron (1999). The data consists of 106 measurements of ratios of strontium isotopes found in fossil shells

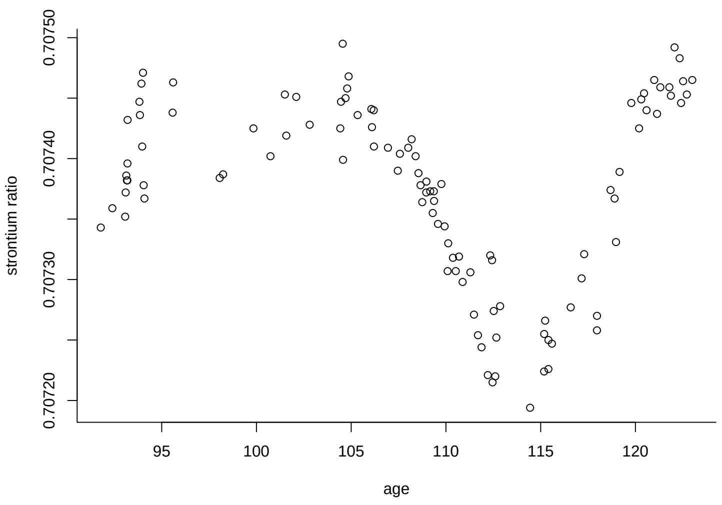


Figure 1: Plot of the fossil data.

and their age and are displayed in Figure 1. These data are stored in the data frame `fossil` in **`SemiPar`**.

We can perform automatic scatterplot smoothing of these data in **`SemiPar`** by specifying

```
data(fossil)
attach(fossil)

fit <- spm(strontium.ratio~f(age))
```

The object **`fit`** is an `R` list containing several pieces of information on the fit. Details are given in Appendix A.

A summary of the fit may be obtained using the function `summary()`:

```
summary(fit)
```

This results in the output:

```
Summary for non-linear components:

           df  spar knots
f(age) 12.14 2.929    25

Note this includes 1 df for the intercept.
```

The fit can be plotted as using:

```
plot(fit)
```

This leads to the plot in Figure 2.

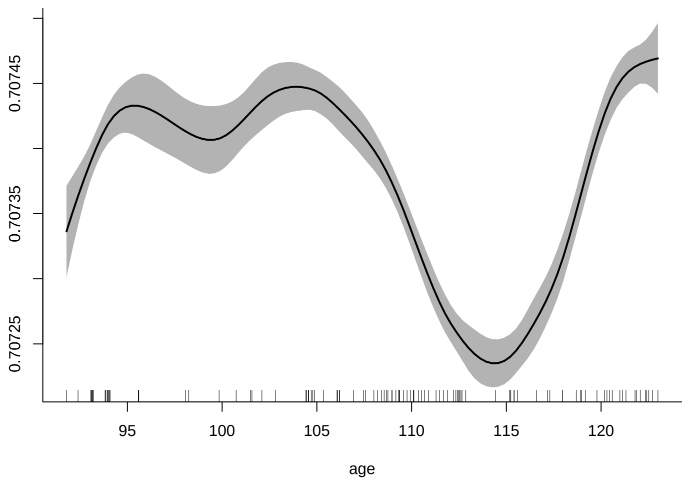


Figure 2: Result of `plot(fit)` for the default fit of (2).

The fit with non-shaded standard error bands can be obtained by specifying

```
plot(fit,shade=FALSE)
```

This leads to the plot in Figure 3.

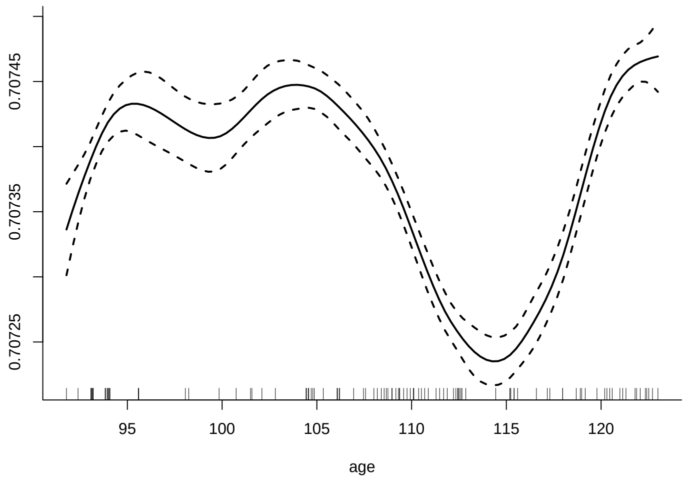


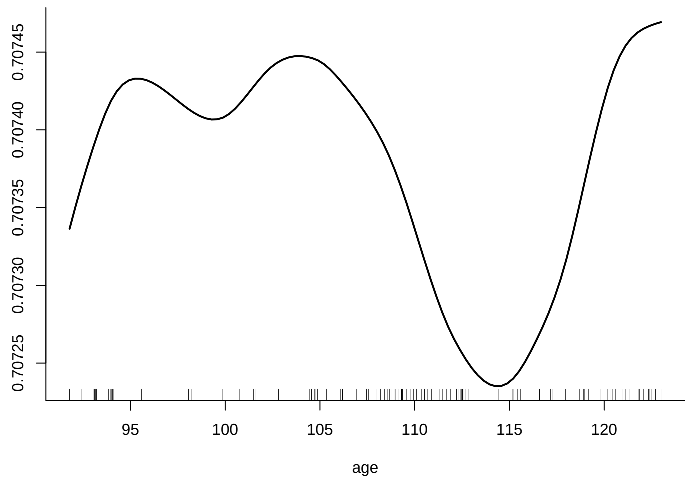
Figure 3: Result of `plot(fit,shade=FALSE)` for the default fit of (3).

To plot without standard error bands we specify:

```
plot(fit,se=FALSE)
```

and this leads to the plot in Figure 4.

Figure 4: Result of `plot(fit,se=FALSE)` for the default fit of (2).



2.2 User specified amount of smoothing

The parameter λ in equation (3) is called the smoothing parameter and if unspecified, is estimated by restricted maximum likelihood (REML) using certain connections between penalised splines and linear mixed models. Details are given in Ruppert, Wand and Carroll (2003) (Chapters 4-5) and Wand (2003). This section illustrates how to override some of the default specifications in **SemiPar**. For instance, to fit a penalised spline regression to the fossil data with a smoothing parameter of 3, we fit:

```
fit <- spm(strontium.ratio~f(age,spar=3))
```

However, a more meaningful measure of the amount of smoothing is the degrees of freedom (e.g. Hastie and Tibshirani, 1990). The number of degrees of freedom corresponding to the REML choice of smoothing parameter for the fossil data is about 12. To fit a penalised spline regression to the fossil data with 21 degrees of freedom, we use:

```
fit <- spm(strontium.ratio~f(age,df=21))
```

This leads to the plot in Figure 5.

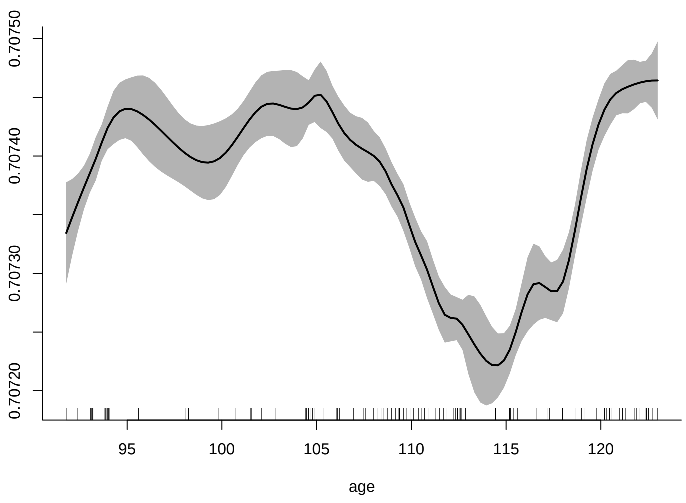


Figure 5: Result of `plot(fit)` for the 21 degrees of freedom fit to the fossil data.

2.3 User specified basis functions

As noted above, the default basis functions correspond to the cubic thin plate splines

$$f(x) = \beta_0 + \beta_1 x + \sum_{k=1}^{K} u_k |x - \kappa_k|^3.$$

Another alternative, supported by **`SemiPar`**, is the truncated polynomial basis functions. These involve f being modelled as a function of the form

$$f(x) = \beta_0 + \beta_1 x + \ldots + \beta_p x^p + \sum_{k=1}^{K} u_k (x - \kappa_k)_+^p \tag{4}$$

Specification of a truncated polynomial basis is done as follows:

```
fit <- spm(strontium.ratio~f(age,basis="trunc.poly"))
```

For radial basis functions the default degree is 3 (cubic), while it is 1 (linear) for truncated polynomials. Basis functions of other degrees can be specified using the `degree` argument as follows:

```
fit <- spm(strontium.ratio~f(age,degree=5))
```

For truncated polynomial basis functions `degree` corresponds to the power p in (4). For thin plate spline basis functions `degree` corresponds to the degree of the radial basis functions, i.e. the `degree` is $2m - 1$ in the notation of (1).

Note that the appropriate multiplier in equation (3) for truncated polynomial bases functions of degree p is λ^{2p} where λ is the smoothing parameter.

Finally, the default choice for knot locations is

$$\kappa_k = \left(\frac{k+1}{K+2}\right) \text{th sample location of the unique } x_i'\text{s}, \quad k = 1, \ldots, K$$

where $K = \max(\frac{n}{4}, 20)$. User-specified knots are also supported in **SemiPar**, or instance, by specifying:

```
knots.fossil <- seq(95,120,length=11)
fit <- spm(strontium.ratio~f(age,knots=knots.fossil))
```

It should be noted here that **SemiPar** does not currently allow use of the "=" sign inside the `spm()` formula specification. For example,

```
fit <- spm(strontium.ratio~f(age,knots=seq(95,120,length=11))
```

is *not* allowable.

2.4 Derivative plots

Better insights into the presence of peaks and valleys in a penalised spline fit can be obtained by looking at a plot of the estimated derivative. Suppose one wants to obtain plots of the first two derivatives of the estimated means for the fossil data. First fit the data with a high degree fit (derivate estimates benefit from smoother fits):

```
fit <- spm(strontium.ratio~f(age,degree=5))
```

Derivative plots are obtained as follows:

```
par(mfrow=c(2,1))
plot(fit,drv=1)
plot(fit,drv=2)
```

and leads to Figure 6.

2.5 User specified plotting parameters

When applied to an `spm()` fit object, additional arguments in `plot()` function can be used to set various plotting specifications. Some examples are:

- Specify the axis labels:

  ```
  plot(fit,xlab="Age of fossils",ylab="Strontium ratio of fossils")
  ```

- Specify axis limits:

  ```
  plot(fit,xlim=c(100,120))
  ```

- Specify some of the line types and widths:

  ```
  plot(fit,lty=4,shade=FALSE,se.lwd=4)
  ```

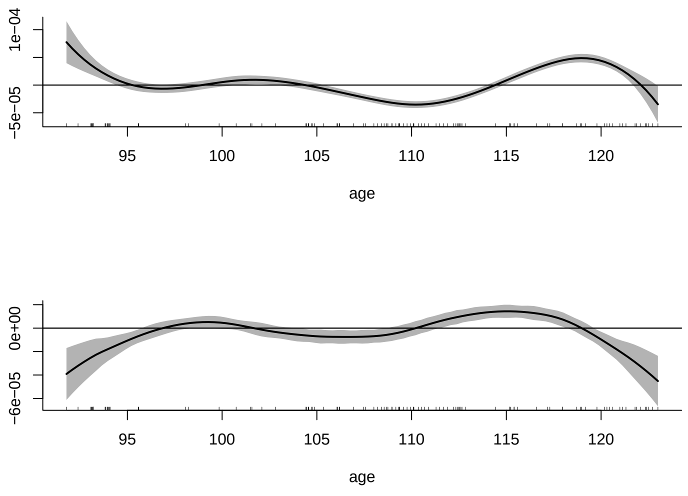


Figure 6: Plots of first and second derivatives for fossil data.

- Specify colours of the lines and shading:

  ```
  plot(fit,col="red",shade.col="cyan")
  ```

The fit in Figure 7 illustrates the result of tweaking several plotting options. The commands that produced this plot are:

```
op <- par(bg="white")
par(bg="honeydew")
plot(fit,ylim=range(strontium.ratio),col="green",
     lwd=5,shade.col="mediumpurple1",rug.col="blue")
points(age,strontium.ratio,col="orange",pch=16)
par(op)
```

Appendix B provides fuller details on plotting parameters.

2.6 Predictions

Predictions for general regions of the predictor space may be made using the `predict()` function. An example for the default fit to the fossil data is:

```
newdata.age <- data.frame(age=c(90,100,110,120,130))
preds <-  predict(fit,newdata=newdata.age,se=TRUE)
print(preds)
```

which yields the list:

```
$fit
[1] 0.7072402 0.7074086 0.7073363 0.7074190 0.7073856
```

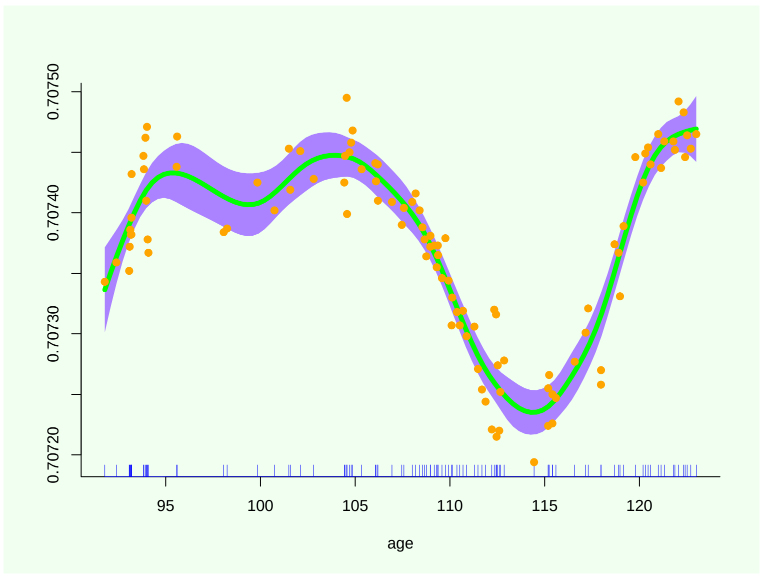


Figure 7: Plot of fit to the fossil data with several plotting parameters tweaked away from their defaults.

```
$se
[1] 4.352286e-05 1.241221e-05 6.764213e-06 8.208272e-06 1.818117e-04
```

The first component of the list are predictions at the specified age values. The second component are corresponding standard errors. The `newdata` argument should be a data frame with names identical to those of the predictor variables.

2.7 Parametric models

Ordinary parametric regression models can be fit using `spm()`. We will illustrate this using the data set `fuel.frame`. First we need to make the data available:

```
data(fuel.frame)
attach(fuel.frame)
```

The straight line regression model

$$\mathsf{E}(\texttt{Fuel}) = \beta_0 + \beta_1\texttt{Weight}$$

may be fit using

```
fit <- spm(Fuel~Weight)
```

whereas the intercept-only model may be fit using

```
fit <- spm(Fuel~1)
```

3 Simple Semiparametric Models

This section considers fitting of simple semiparametric models; i.e. regression models with a non-parametric component in one predictor and a parametric component in another predictor. We shall illustrate fitting such models in **SemiPar** using a data set on yield of onions in two locations in South Australia (Ratkowsky, 1983; Young and Bowman, 1995). The data are part of **SemiPar** and are stored in the object `onions`.

Make data available to current session:

```
data(onions)
attach(onions)
log.yield <- log(yield)
```

The simple semiparametric regression model

$$\mathsf{E}(\texttt{log.yield}_i) = \beta\,\texttt{location}_i + f(\texttt{dens}_i)$$

may be fit using `spm()` as follows:

```
fit <- spm(log.yield~location+f(dens))
```

A summary of the fit may be obtained using the function `summary()`:

```
summary(fit)
```

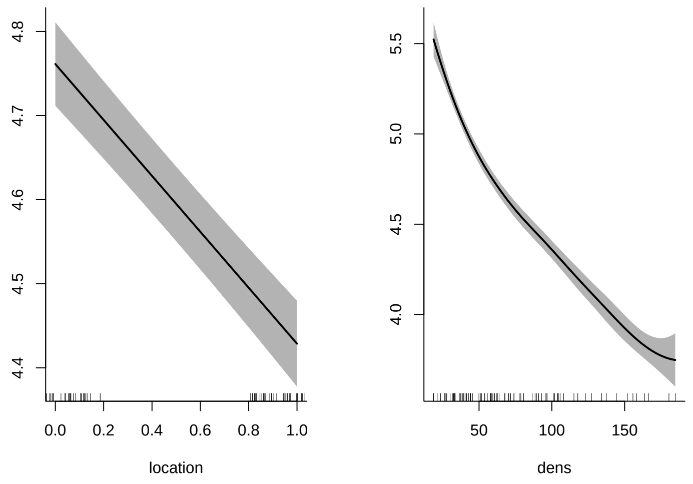
This results in the output:

```
Summary for linear components:

             coef      se  ratio p-value
intercept  5.3880 0.24230  22.24       0
location  -0.3325 0.02388 -13.92       0




Summary for non-linear components:

           df  spar knots
f(dens) 4.213 63.02    17
```

Also, the commands

```
par(mfrow=c(1,2))
plot(fit,jitter.rug=TRUE)
```

Figure 8: Plot of simple semiparametric fit to the onions data.



lead to the plots in Figure 8.

Default basis and smoothing specifications for the nonparametric component can be overridden by user specified options just as for the case of scatterplot smoothing.

4 Additive models

Additive models are an extension of simple semiparametric models that allow for several predictors to have a nonparametric smooth functional form. We will illustrate the fitting of additive models in **SemiPar** using data on atmospheric ozone concentration and meteorology in the Los Angeles basin. See Breiman and Friedman (1985) for details. The response is daily ozone concentration (`ozone.level`) and the predictors are:

`daggett.pressure.gradient`	pressure gradient at Daggett in mmHg
`inversion.base.height`	inversion base height, in feet
`inversion.base.temp`	inversion based temperature, in degrees Fahrenheit

In **SemiPar** these data are stored in the data frame `calif.air.poll`.

Make data available to current session:

```
data(calif.air.poll)
attach(calif.air.poll)
```

The additive model

$$\mathsf{E}(\texttt{ozone.level}_i) = f_1(\texttt{daggett.pressure.gradient}_i) + f_2(\texttt{inversion.base.height}_i)$$

$$+f_3(\texttt{inversion.base.temp}_i)$$

may be fit using the command:

```
fit <- spm(ozone.level ~ f(daggett.pressure.gradient)+
                         f(inversion.base.height) +
                         f(inversion.base.temp))
```

A summary of the fit can be obtained by using the `summary` function:

```
summary(fit)
```

This results in the following output:

```
Summary for non-linear components:
                               df    spar knots
f(daggett.pressure.gradient) 4.697   88.80    31
   f(inversion.base.height)  4.198 2741.00    39
     f(inversion.base.temp)  3.248   57.98    38
```

For a display of the additive components, we specify:

```
par(mfrow=c(2,2))
plot(fit)
```

This results in the plot in Figure 9.

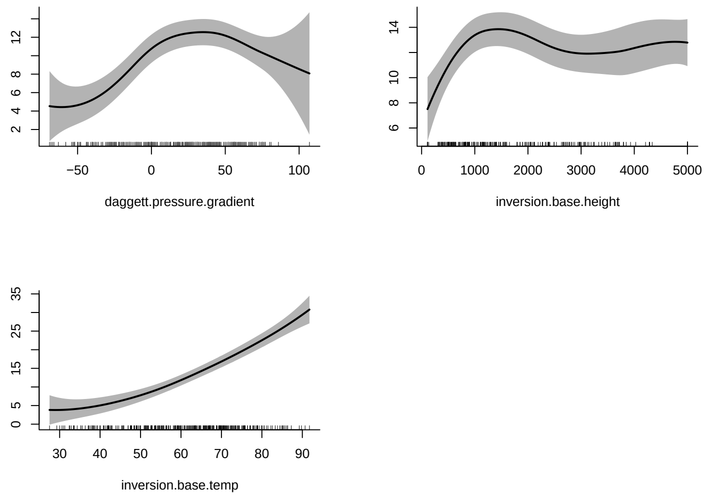


Figure 9: Result of `plot(fit)` for the fit to the Californian air pollution data.

Plots from additive model fits may be customised using the additional arguments in the call to `plot()` in `spm()`. The following example provides illustration:

```
par(mfrow=c(2,2))
op <- par(bg="white")
par(bg="darkseagreen")
plot(fit,shade=FALSE,col=c("red","orange","gold"),
     lwd=rep(5,3),se.lwd=rep(3,3),
     se.col=c("greenyellow","blue","purple"),
     rug.col=c("navy","deeppink","darkorange"),
     xlim=list(lower=c(-50,0,30),upper=c(80,4200,90)),
     xlab=c("Daggett pressure gradient","Inversion base height",
            "Inversion based temperature"),
     ylab=rep("Contribution to mean ozone level",3))
par(op)
```

The resulting plot is shown in Figure 10.

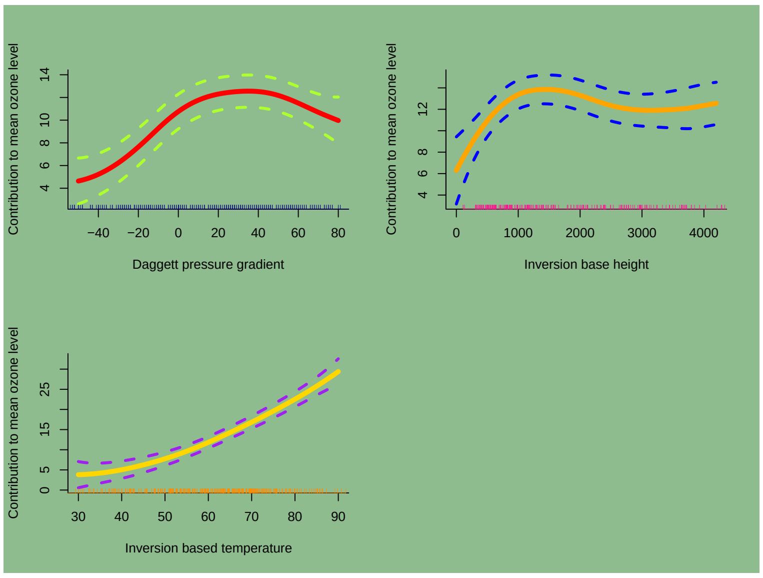


Figure 10: Result when customised example of `plot()` is used to display the fit to the Californian air pollution data.

The user can also specify knots and degrees of freedom values. For degrees of freedom this needs to be done via the `adf` (approximate degrees of freedom) argument. An example is:

```
fit <- spm(ozone.level ~ f(daggett.pressure.gradient,adf=6)+
                          f(inversion.base.height,adf=4) +
                          f(inversion.base.temp,adf=9))
```

Note that only approximate degrees of freedom can be pre-specified for additive models since it is very computationally expensive to find smoothing parameters that give an exact pre-specified degrees of freedom. The approximation is based on individual univariate fits. Also, the intercept is not included in the approximate degrees of freedom — so the total approximate degrees of freedom of the above model is 1+6+4+9=20.

5 Additive mixed models

The mixed model representation of penalised splines allows for a seamless fusion of random effects models for longitudinal data and smoothing. The simplest such model is the *additive mixed model*, and is supported by **`SemiPar`**. For illustration we shall use data on the growth of Sitka spruces displayed in Figure 1.3 of Diggle, Liang and Zeger (1995). The data consists of growth measurements of 79 trees over two seasons: 54 trees were grown in an ozone-enriched atmosphere while the remaining 25 comprise a control group.

A useful additive mixed model for these data is:

$$\log(\texttt{size})_{ij} = U_i + \beta\,\texttt{ozone}_i + f(\texttt{days}_{ij}) + \varepsilon_{ij}, \quad 1 \leqslant j \leqslant n_i, \quad 1 \leqslant i \leqslant m \qquad (5)$$

where

$$U_i \overset{\text{ind.}}{\sim} N(0, \sigma_u^2)$$

are random intercepts for each tree and the ε_{ij} are random errors.
First, we make data available to current session:

```
data(sitka)
attach(sitka)
```

We then obtain a fit of (5) with REML choice of degrees of freedom for f and REML estimation of σ_u^2:

```
fit <- spm(log.size~ozone+f(days),random=~1,group=id.num)
```

To view the summary of the fit:

```
summary(fit)
```

This leads to the following output:

```
Summary for linear components:

               coef      se   ratio p-value
intercept   8.4230 0.1743  48.340  0.0000
ozone      -0.3006 0.1493  -2.013  0.0444




Summary for non-linear components:

            df spar knots
f(days) 2.998 97.9     2


Summary for random intercept component:
```

```
                    df
random intercept 76.35
```

We can plot the fit as before:

```
par(mfrow=c(1,2))
plot(fit)
```

The result is shown in Figure 11.

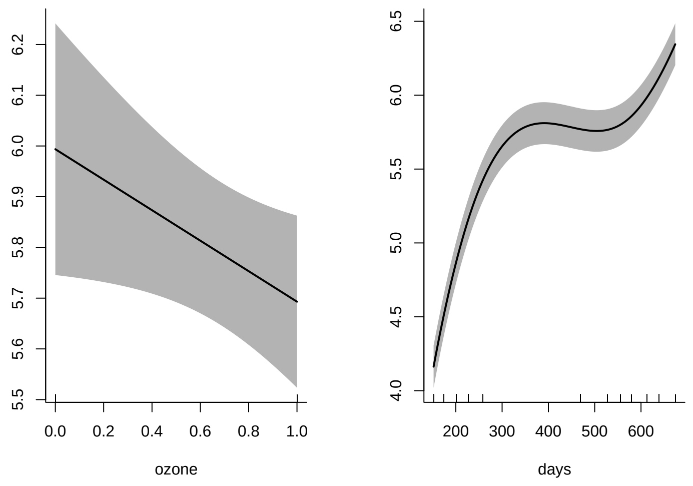


Figure 11: Result of `plot(fit)` for the default fit of (5).

6 Bivariate smoothing

One of **SemiPar**'s more attractive features is its handling of the bivariate smoothing problem. As shown in Section 7 **SemiPar** also allows for bivariate smooths to be incorporated into additive models.

In this section we consider the situation where data are available on a response y and bivariate predictor $\mathbf{x} \in \mathbb{R}^2$ and we want to fit

$$y_i = f(\mathbf{x}_i) + \varepsilon_i \tag{6}$$

where f is a smooth bivariate function. In many applications $\mathbf{x}_i$ represents a geographical location, but may also represent two continuous predictors for which additivity is not reasonably assumed.

Estimation of f in (6) is done by using radial basis functions approximation; with the family of basis functions corresponding to the thin plate spline family as summarised by Wahba (1990) and Nychka (2000). In the notation given there the case $m = 1$ corresponds to

$$f(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}_1^T \mathbf{x} + \sum_{k=1}^{K} u_k \|\mathbf{x} - \boldsymbol{\kappa}_k\|^2 \log \|\mathbf{x} - \boldsymbol{\kappa}_k\|. \tag{7}$$

Here $\boldsymbol{\kappa}_1, \ldots, \boldsymbol{\kappa}_K \in \mathbb{R}^2$ are a set of knots that "cover" the space of the $\mathbf{x}_i$. Default knots in **SemiPar** are selected using the *clara* algorithm of Kaufman and Rousseeuw (1990). This is available in the R package cluster.

We shall use data on the location of scallop catches in the Atlantic continental shelf off Long Island, New York, USA, to illustrate bivariate smoothing (e.g. Ecker and Heltshe, 1994). These data are stored in the scallop data frame in **SemiPar**.

Suppose that the data are made available to the current session as follows:

```
data(scallop)
attach(scallop)
log.catch <- log(tot.catch+1)
```

Then a fit with default knot and degrees of freedom is obtained via:

```
fit <- spm(log.catch~f(longitude,latitude))
```

When default knot choice is specified a figure showing knot location is sent to the screen. This is to help ensure that the default knots do a good job of 'filling up the space' of the biviariate predictor data. Figure 12 shows these default knots for the scallop data. Here it is seen that the default knots fill the space quite satisfactorily.

To view an image of the fitted values, we specify:

```
plot(fit)
```

With this command the user will be prompted to specify a polygonal boundary — the region for which pixels in the image plot are switched on. It is generally recommended that the boundary corresponds roughly to the region of highest density of the bivariate data. For a boundary chosen to correspond to the longitude/latitude data the plot in Figure (13) results. A summary of the bivariate fit can be viewed as before, by specifying:

```
summary(fit)
```

This leads to the following output:

```
Summary for non-linear components:


                          df   spar knots
f(longitude,latitude) 25.12 0.2904    37
```

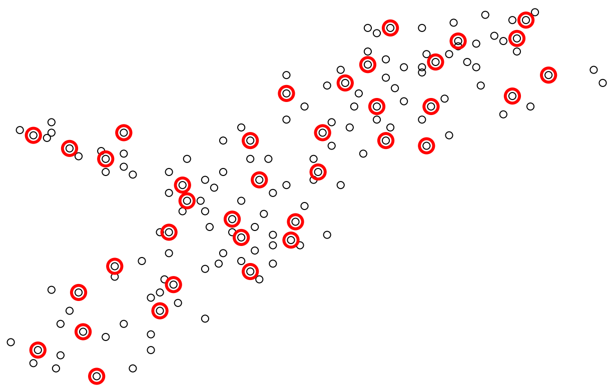


Figure 12: Default knots for the scallop data.

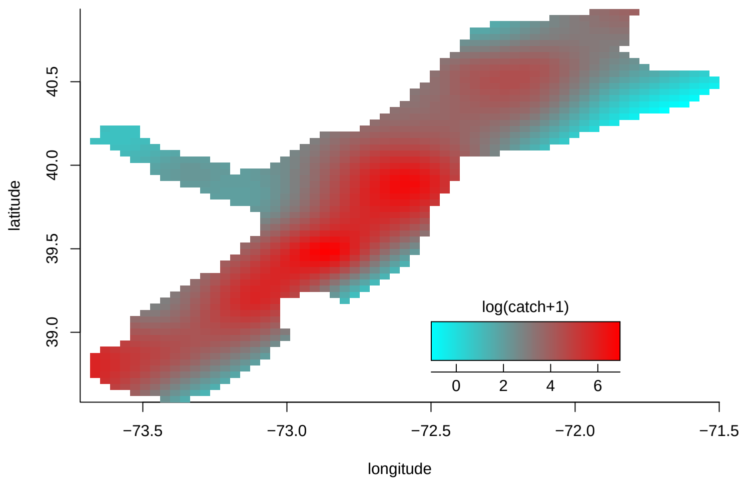


Figure 13: Result of `plot(fit)` for the default fit of (6).

If either the knots or boundary polygon are not specified then the user is prompted to save this information in a file. This is advisable for data sets that will be re-analysed since knot choice and boundary specification is time-consuming. Suppose that the knots are saved to the file `scallop.knots` and the boundary polygon vertices are saved to the file `scallop.bdry`. Then Figure 13 can be re-produced using the commands:

```
scp.knots <- read.table("scallop.knots")
scp.bdry <- read.table("scallop.bdry")
```

```
fit <- spm(log.catch~f(longitude,latitude,knots=scp.knots))
plot(fit,bdry=scp.bdry,image.zlab="log(catch+1)")
```

As before, the parameter λ in equation (3) is called the smoothing parameter and if unspecified, it is estimated by Restricted Maximum Likelihood using connections between penalised splines and linear mixed models. To fit a penalised spline regression to the scallop data with a smoothing parameter of 3, we fit:

```
fit <- spm(log.catch~f(longitude,latitude,spar=3))
```

The REML degrees of freedom corresponding to the REML choice of smoothing parameter for the scallop data was 9.7. To fit a penalised spline regression to the scallop data with say, 35 degrees of freedom in the predictor, we fit:

```
fit <- spm(log.catch~f(longitude,latitude, adf=35))
```

This leads to the plot in Figure 14.

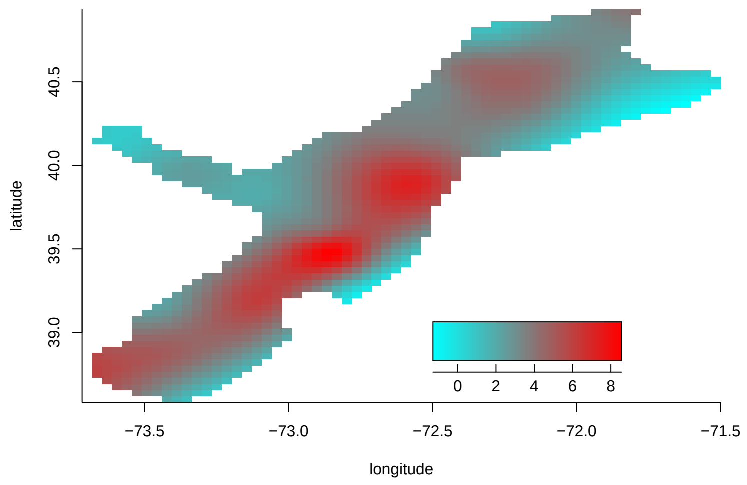


Figure 14: Result of `plot(fit)` for a 35 degrees of freedom fit to the scallop data.

If the bivariate knots are not specified then, as mentioned above, default knots are chosen using a the clara algorithm. The default number of knots is

$$K = \max\{10, \min(50, \text{round}(n/4))\}.$$

Depending on the value of n and K, the knot selection algorithm can be quite slow. Therefore it is recommended that these be saved to a file and, for future analyses of the same data, the knots be inputted in the call to `spm()`.

7 Geoadditive models

Geoadditive models are described in Kammann and Wand (2003). We will provide illustration via the copper data from Clark and Harper (2000). They consist of measurements on the grade of copper from a simulation based on a stockpile of mined material in the former Soviet Union. Each grade measurement is accompanied by a three-dimensional location vector; denoted $(\texttt{xcoord}, \texttt{ycoord}, \texttt{zcoord})$. The natural full model for these data is the three-dimensional smoothing model

$$\texttt{grade}_i = f(\texttt{xcoord}, \texttt{ycoord}_i, \texttt{zcoord}_i) + \varepsilon_i.$$

For the purpose of illustrating geoadditive models we will fit

$$\texttt{grade}_i = f_1(\texttt{xcoord}_i, \texttt{ycoord}_i) + f_2(\texttt{zcoord}_i) + \varepsilon_i. \tag{8}$$

First make the `copper` data available to the current session:

```
data(copper)
attach(copper)
```

Bivariate knot choice here is a little delicate since the default number of knots is possibly too low. Therefore we will get the bivariate knot selection out of the way and save them in a file:

```
copper.knots <- default.knots.2D(xcoord,ycoord,num.knots=20)
```

While we're at it, do the same for the boundary file (needed for effective plotting of the bivariate surface fit):

```
copper.bdry <- default.bdry(xcoord,ycoord)
write(t(copper.bdry),"copper.bdry",ncol=2)
```

Default knot selection is inadequate for `zcoord`, due to the small number of unique values of this variable. Therefore, specify knots for `zcoord`:

```
knots.z <- seq(80,120,by=5)
```

Fit (8) with REML choice of degrees of freedom and omission of cases with a missing grade value:

```
copper.knots <- read.table("copper.knots")
fit <- spm(grade~f(xcoord,ycoord,knots=copper.knots)
           +f(zcoord,knots=knots.z),omit.missing=TRUE)
```

View the summary of the fit:

```
summary(fit)
```

The following output results:

```
Summary for non-linear components:

                     df   spar knots
       f(zcoord)  5.384 19.3       9
f(xcoord,ycoord) 12.690 325.4     20
```

The high number of degrees of freedom in each shows that REML has chosen a non-linear zcoordeffect and non-linear surface for $(\texttt{xcoord}, \texttt{ycoord})$ as well.

Finally, we display the fit graphically. Here we will choose a finer mesh for the image plot and position the legend near the lower left corner:

```
copper.bdry <- read.table("copper.bdry")
par(mfrow=c(1,1))
plot(fit,bdry=copper.bdry,image.grid.size=c(100,100),
     leg.loc=c(300,300))
```

The result is shown in Figure 15.

8 Generalised responses

The current release of **SemiPar** accommodates binary and count response data via the binomial and Poisson models, respectively, with canonical link functions. For example, if the data are (x_i, y_i), $1 \leqslant i \leqslant n$, where $y_i \in \{0, 1\}$ is binary then model

$$P(y_i = 1|x_i) = \frac{\exp\{f(x_i)\}}{1 + \exp\{f(x_i)\}}, \tag{9}$$

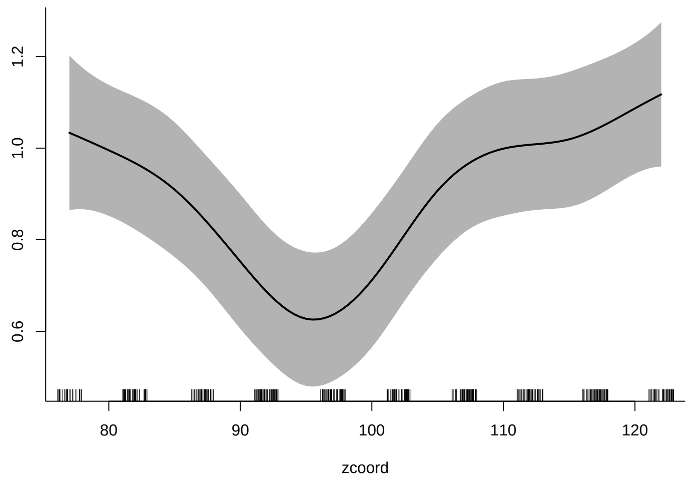
with $f(x)$ having a basis function expansion such as (1) and random effects subject to (2), is allowable. If no smoothing parameter or degrees of freedom value is included then the smoothing parameters are chosen using *penalised likelihood* via the function glmmPQL() from the MASS package.

We will illustrate the fitting of this model for data on trade union membership (1=member, 0=non-member) and wages; part of the dataset trade.union in **SemiPar** . These data are displayed in Figure 16.
The data can be fitted using:

```
data(trade.union)
attach(trade.union)
union.member <- union.member[-171]
wage <- wage[-171]
fit <- spm(union.member~f(wage),family="binomial")
```

A plot of the result can be obtained using

```
plot(fit)
```



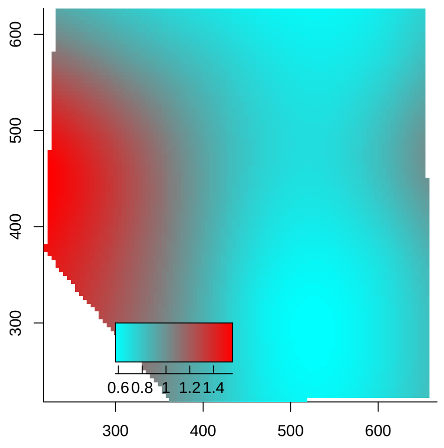


Figure 15: Result of `plot(fit)` for the default fit of (8).

The result is shown in Figure 17.

The vertical axis corresponds to the estimate of $f(x)$ on the inverse link scale. In the binary response case this is

$$\text{logit}\{\widehat{f}(x)\} \equiv \log[\widehat{f}(x)/\{1 - \widehat{f}(x)\}].$$

Embellishments to the plot can be made through the advice given in Section 2.5.

Each of the more complicated semiparametric regression models described in Sections 3 to 7 can also be fit for generalised responses provided that the user

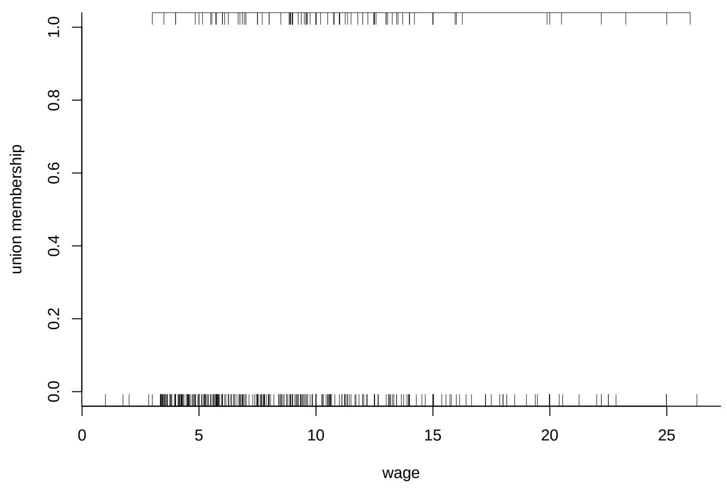


Figure 16: A binary response regression data set with presence of trade union membership versus wage from the data set `trade.union`.

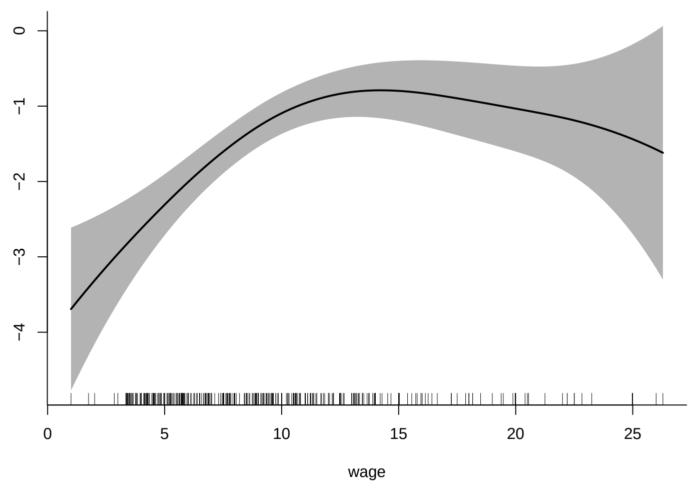


Figure 17: Result of `plot(fit)` applied to fit of model (9) for the union membership and wage.

specify the amount of smoothing. For example, the generalised additive model

$$\begin{aligned}&\text{logit}\{P(\texttt{union.membership} = 1|\texttt{wage}, \texttt{years.educ}, \texttt{age}, \texttt{female}, \texttt{white}, \texttt{south})\}\\&\quad = \beta_0 + f_1(\texttt{wage}) + f_2(\texttt{years.educ}) + f_3(\texttt{age}) + \beta_4\,\texttt{female} + \beta_5\,\texttt{white} + \beta_6\,\texttt{south}\end{aligned}$$

can be fit (to the data with high leverage observation 171 omitted) using:

```
age <- age[-171]
years.educ <- years.educ[-171]
```

```
female <- female[-171]
race <- race[-171]
white <- as.numeric(race==3)
south <- south[-171]

knots.years.educ <- seq(6.5,17.5,by=1)
fit <- spm(union.member~f(wage)+f(years.educ,knots=knots.years.educ)
            +f(age)+female+white+south,family="binomial")
```

The fit shown in Figure 18 is a result of the commands:

```
summary(fit)

Summary for linear components:

             coef     se  ratio p-value
intercept -0.8545 0.8355 -1.023  0.3066
female    -0.7027 0.2653 -2.649  0.0082
white     -0.7185 0.2958 -2.429  0.0153
south     -0.5373 0.2926 -1.837  0.0665




Summary for non-linear components:

                 df   spar knots
f(wage)       2.661  16.59    38
f(years.educ) 1.006 124.90    12
f(age)        1.001 808.80    10
```

The above output indicates that the effects of `years.educ` and `age` are fairly linear so a more appropriate (simpler) model is obtained through the command:

```
fit <- spm(union.member~f(wage)+years.educ+age+female
           +white+south,family="binomial")
par(mfrow=c(3,2))
plot(fit,jitter.rug=TRUE)
summary(fit)
```

This has summary:

```
Summary for linear components:

                coef      se  ratio p-value
intercept  -0.85560 0.83540 -1.024  0.3058
years.educ -0.08196 0.05068 -1.617  0.1060
age         0.01829 0.01101  1.661  0.0969
```

```
female      -0.70260 0.26530 -2.648  0.0081
white       -0.71830 0.29580 -2.428  0.0152
south       -0.53730 0.29260 -1.836  0.0664


Summary for non-linear components:

           df  spar knots
f(wage) 2.661 16.59    38
```

showing that each of the linear variables are statistically significant to varying degrees.

The plot in Figure 18 is obtained via the command

```
plot(fit,jitter.rug=TRUE)
```

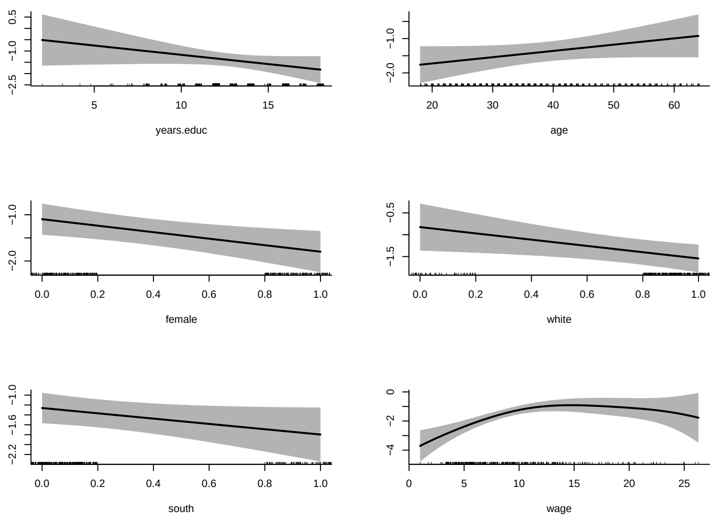


Figure 18: Plots for the generalised model with `wage` non-linear but all other variables linear.

Appendix A: Details on `spm()` objects

The `fit` object returned by `spm()` is a list with three components:

```
[1] "fit"  "info" "aux"
```

For the default value of `family` the object `fit$fit` has the same structure as objects from the `lme()` function in the `nlme` package. For `family` set to either `"binomial"` or `"poisson"` then `fit$fit` has the same structure as objects from the `glmmPQL()` function in the `MASS` package.

The object `fit$aux` is a list containing estimated covariance matrices of the fixed and random effects (`$cov.mat`). It also contains estimated variances of the random effects (`$random.var`) and residual errors (`$resid.var`). The list also contains approximate degrees of freedom for each additive component (`$df`) and the overall degrees of freedom for the fit (`$df.fit`).

The object `fit$info` is a list containing information about the model set-up: e.g. design matrices, knots, type of basis functions and polynomial degree.

The `names()` function can be used to examine all compnents and sub-components of `fit`.

Appendix B: Details on Plot Parameters

Table 2 describe all plotting parameters and their default values for calls to `plot()` on an `spm()` fit object.

For models with several curves many of the parameters in Table 2 can be specified for each curve. For example, if there are four curves in the `spm()` object `fit` then a possible specification is:

```
plot(fit,lty=c(1,3,5,6),lwd=rep(3,4))
```

Appendix C: Trouble Shooting

Due to resource limitations **`SemiPar`** does not have as many safeguards as professional software packages. Below we list some tips for avoiding "crashes" in calls to `spm()`.

- If the default value of `family` is used then the methodology is based on an approximate assumption of Gaussianity of the response variable. If this assumption is heavily violated (e.g. gross outliers, strong skewness) then the fitting algorithm could be adversely affected. Outliers should be treated with caution and removed if justifiable. Transformations for strongly skewed response variables are worth considering.

parameter	description	default (for all curves)
`plot.it`	flag for plotting curve estimate	`TRUE`
`drv`	order of derivative to plot	`0`
`se`	flag for adding pointwise ±2 std. err. bands	`TRUE`
`shade`	flag for using shading in std. err. bands	`TRUE`
`bty`	type of box drawn around plots	`"l"`
`main`	main title of plot	`""`
`xlab`	x-axis labels	variable name
`ylab`	y-axis labels	`""`
`xlim`	x-axis limits	range of x-values
`ylim`	y-axis limits	curve vertical ranges
`grid.size`	number of grid points	`101`
`lty`	line type for curve estimate	`1`
`lwd`	line width for curve estimate	`2`
`col`	line colour for curve estimate	`black`
`se.lty`	line type for std. err. bands	`2`
`se.lwd`	line width for std. err. bands	`2`
`se.col`	line colour for std. err. bands	`black`
`shade.col`	colour of shaded std. err. bands	`grey70`
`rug.col`	colour of rug representation of x-values	`black`
`jitter.rug`	flag for jittering of rug representation	`FALSE`
`zero.line`	flag for adding zero line to derivative plots	`TRUE`
`plot.image`	flag for image of bivariate surface estimate	`TRUE`
`image.col`	colour vector used in image plot	`cyan` to `red`
`image.bg`	background of image	`white`
`image.bty`	type of box drawn around image	`"l"`
`image.main`	main title of image	`""`
`image.xlab`	x-axis label of image	x-variable name
`image.ylab`	y-axis label of image	y-variable name
`image.zlab`	label for z variable	`""`
`image.xlim`	x-axis limits of image	range of x-values
`image.ylim`	y-axis limits of image	range of y-values
`image.zlim`	range of z-values in image	range of z-values
`image.grid.size`	two-component array of grid-sizes for image	`c(64,64)`
`bdry`	two-column matrix of polygonal boundary for image	dynamic user-specification
`add.legend`	flag for image legend	`TRUE`
`leg.loc`	location of top-left corner of legend	lower left region
`leg.dim`	dimension of legend in x- and y-variable units	30%$\times$10% of plot box
`image.zlab.col`	colour of legend label	`black`

Table 2: Details on plot parameters for `spm()` fit objects.

- Only continuous variables should be used as non-linear predictors. In computing terms, a continous variable is one with many non-unique values. If the variable `x` has only 4 unique values then the command:

  ```
  fit <- spm(y f(x))
  ```

  is prone to error. For predictor variables with moderate numbers of unique values (around 10), it is worth experimenting with the choice of knots.

- As with all regression models, predictor values with high leverage; i.e. lying a long way from the main body of the data, can have an undue influence on the fit and should be treated cautiously.
- For bivariate fits a graphical check of the knots relative to the bivariate predictors is recommended (this is provided by `default.knots()`). If the knots seem too sparse or dense relative to the data then other knot choices should be considered.
- Be careful to specify the parameters in exactly the correct way. For example,in

  ```
  fit <- spm(log.catch~f(longitude,latitude,knots=scp.knots))
  ```

  it is important that `scp.knots` is a two-column array. If, instead, `scp.knots` was a two-component list then the command would fail.
- Make sure that all variables in the call to `spm()` actually exist as arrays in the current session.
- If `family` is specified to be `"binomial"` or `"poisson"` and no smoothing parameter or degrees of freedom is specified then the function `glmmPQL()` from the `MASS` package is used for fitting. We have experienced mixed success with this function. It is prone to crashing more often then `lme()` as used for Gaussian response models. Until this problem is resolved you may have to work with user-specified degrees of freedom values. The function `gam()` in the `mgcv` may also overcome such problems.

References

Bralower, T.J., Fullagar, P.D., Paull, C.K., Dwyer, G.S. and Leckie, R.M. (1997). Mid-cretaceous strontium-isotope stratigraphy of deep-sea sections. *Geological Society of America Bulletin*, **109**, 1421–1442.

Breiman, L. and Friedman, J. (1985). Estimating optimal transformations for multiple regression and correlation (with discussion). *Journal of the American Statistical Association*, **80**, 580–619.

Chaudhuri, P. & Marron, J.S. (1999). SiZer for exploration of structures in curves. *Journal of the American Statistical Association*, **94**, 807–823.

Clark, I. and Harper, W.V. (2000). *Practical Geostatistics 2000.* Columbus, Ohio: Ecosse North America Llc.

Diggle, P., Liang, K.-L. and Zeger, S. (1995). *Analysis of Longitudinal Data*. Oxford: Oxford University Press.

Ecker, M.D. and Heltshe, J.F. (1994). Geostatistical estimates of scallop abundance. In *Case Studies in Biometry.* Lange, N., Ryan, L., Billard, L., Brillinger, D., Conquest, L. and Greenhouse, J. (eds.) New York: John Wiley & Sons, 107–124.

Eilers, P.H.C. & Marx, B.D. (1996). Flexible smoothing with B-splines and penalties (with discussion). *Statistical Science*, **11**, 89–121.

French, J.L., Kammann, E.E. & Wand, M.P. (2001). Comment on paper by Ke and Wang. *Journal of the American Statistical Association*, **96**, 1285–1288.

Godtliebsen, F., Marron, J.S. & Chaudhuri, P. (2000). Statistical significance of features in digital images. *Image and Vision Computing*, **13**, 1093–1104.

Hastie, T.J. and Tibshirani, R.J. (1990). *Generalized Additive Models*. London: Chapman and Hall.

Kammann, E.E. & Wand, M.P. (2003). Geoadditive models. *Applied Statistics*, **52**, 1–18.

Kaufman, L. and Rousseeuw, P.J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis.* New York: Wiley.

Kent, J.T. and Mardia, K.V. (1994). The link between kriging and thin plate splines. In *Probability, Statistics and Optimization: a Tribute to Peter Whittle* (ed. F.P. Kelly), pp. 325–339. Chichester: John Wiley & Sons.

Nychka, D.W. (2000). Spatial process estimates as smoothers. In *Smoothing and Regression* (M. Schimek, ed.). Heidelberg: Springer-Verlag.

Nychka, D., Haaland, P., O'Connell, M., Ellner, S. (1998). FUNFITS, data analysis and statistical tools for estimating functions. In *Case Studies in Environmental Statistics* (D. Nychka, W.W. Piegorsch, L.H. Cox, eds.), New York: Springer-Verlag, 159–179.

Nychka, D. & Saltzman, N. (1998). Design of Air Quality Monitoring Networks. In *Case Studies in Environmental Statistics* (D. Nychka, Cox, L., Piegorsch, W. eds.), Lecture Notes in Statistics, Springer-Verlag, 51–76.

O'Connell, M.A. and Wolfinger, R.D. (1997). Spatial regression models, response surfaces, and process optimization. *Journal of Computational and Graphical Statistics*, **6**, 224–241.

Pinheiro, J.C. and Bates, D.M. (2000). *Mixed-Effects Models in S and S-PLUS*. New York: Springer.

Ratkowsky, D. A. (1983). *Nonlinear Regression Modeling: A Unified Practical Approach.* New York: Marcel Dekker.

Robinson, G.K. (1991). That BLUP is a good thing: the estimation of random effects. *Statistical Science*, **6**, 15–51.

Ruppert, D. & Carroll, R.J. (2000). Spatially-adaptive penalties for spline fitting. *Australian and New Zealand Journal of Statistics*, **42**, 205–224.

Ruppert, D., Wand, M. P. & Carroll, R.J. (2000). *Semiparametric Regression*. New York: Cambridge University Press.

Venables, W.N. and Ripley, B.D. (1997). *Modern Applied Statistics with S-PLUS*. New York: Springer.

Wahba, G. (1990). *Spline Models for Observational Data.* Philadelphia: SIAM.

Wand, M. P. (2003). Smoothing and mixed models. *Computational Statistics*, **18**, 223–249.

Young, S. G. and Bowman, A. W. (1995). Non-parametric analysis of covariance. *Biometrics*, **51**, 920–931.