User's Manual for BMMP S-PLUS/R Software (version 1.3)

Iain Pardoe [*]

Charles H. Lundquist College of Business

University of Oregon

September 15, 2002

Abstract

This manual describes software for S-PLUS and R that constructs Bayes marginal model plots Pardoe (2001), a graphical method for checking the fit of a regression model. It describes version 1.3 of the software, which currently works for linear models, additive models and generalized linear models.

---

[*]Iain Pardoe is Assistant Professor, Department of Decision Sciences, Charles H. Lundquist College of Business, University of Oregon, Eugene, OR 97403-1208 (E-mail: ipardoe@lcbmail.uoregon.edu).

Contents

1 Introduction

This manual describes software for S-PLUS and R that can produce the analyses that are described in Cook and Pardoe (2000); Pardoe (2001); Pardoe and Cook (2002). The software is available at:

> http://lcb1.uoregon.edu/ipardoe/research/bmmpsoft.htm

Special features of the software include the following:

- it works with linear models, additive models and generalized linear models

- up to six plots can be displayed at a time in one window

- random linear combinations of model predictors can be calculated automatically for use as horizontal axis plotting quantities

- smoothing parameters for the smooths used in the plots can be selected interactively, pre-specified or set using default values

- subsets of the data can be specified, for example if a dataset contains males and females, then plots can be constructed for each gender separately

- either "Trellis" graphics (Becker and Cleveland, 1996) or traditional graphics can be used (in R, only traditional graphics are available)

- the user can choose the smoothing method used to produce the analyses, currently either smoothing splines (see Hastie and Tibshirani, 1990), "loess" (see Cleveland and Devlin, 1988), kernel smooths (see Silverman, 1986) or Friedman's "super smoother" (Friedman, 1984)

- missing values and weights are handled automatically

- binary responses are automatically jittered to aid visualization of data density

Section 2 describes how to install the software, while Section 3 provides a brief description of the functions included. A detailed guide on the use of the functions as well as some examples are included in Section 4. Refer to Section 5 if you encounter any confusing error messages or the software does not behave as expected. Finally, Sections 6 and 7 provide brief outlines on the use of supplementary software that can be used to provide the inputs to this software.

2 Installation

All the functions needed are in zipped files available at:

> http://lcb1.uoregon.edu/ipardoe/research/bmmpsoft.htm

Download either `bmmp**.s.gz` or `bmmp**.s.zip` for use with S-PLUS and either

`bmmp**.R.gz` or `bmmp**.R.zip` for use with R. The functions have been tested on UNIX, Linux and Windows platforms, but no guarantee can be made for their working satisfactorily in all situations. In particular, regression smoothing can sometimes fail for various combinations of smoothing method, smoothing parameter, dataset, and platform. Try other smoothing methods and smoothing parameters to see if this fixes the problem.

Unzip the downloaded file (UNIX/Linux users will probably want the `gz` files, Windows users the `zip` files) and use `source()` to load the functions. To make the functions always available without cluttering up your workspace, do the following:

- Create a subdirectory wherever you usually start S-PLUS (or R) called `bmmp` (say). For S-PLUS 5.x or 6.x on UNIX/Linux type `SPlus CHAPTER` in this subdirectory, otherwise just go on to the next step.

- Start S-PLUS (or R) in the subdirectory.

- Enter `source("pathname/bmmp**.s")` (where "pathname" is wherever you put the unzipped file from above) or `source("pathname/bmmp**.R")` and then quit (remembering to save the workspace image in R). For S-PLUS 6.x on Windows instead of `source()` use `createChapter(where="pathname")` and `sourceChapter(where="pathname")`.

- Start S-PLUS (or R) in your usual workspace and create a `.First` function to attach the functions in `bmmp`, for example:
  ```
  .First <- function() attach("pathname/bmmp")
  ```
  (S-PLUS for UNIX/Linux)
  ```
  .First <- function() attach("C:\\pathname\\bmmp\\_Data")
  ```
  (S-PLUS for Windows)
  ```
  .First <- function() attach("pathname/bmmp/.RData")
  ```
  (R)
  Every time you start up from now on, the functions will be available for use.

A script (`umscript`) containing the commands for running the examples in this chapter is also available at the software web-site.

3   Brief description of the functions

The software consists of the following seventeen (S-PLUS) or ten (R) functions (the starred functions are *not* available in R):

bmmp Construct a Bayes marginal model plot for a fitted model which can be a `lm`, `gam` or `glm` object

bmmp.gam* Support function for `bmmp` that works with `gam` objects

bmmp.glm Support function for `bmmp` that works with `glm` objects

bmmp.lm Support function for `bmmp` that works with `lm` objects

bmmp.default Support function for `bmmp` that does the final calculations and plotting

panel.sp* Panel function used with `mmp` and `bmmp`

panel.mmp* Panel function used with `mmp` and `bmmp`

panel.bmmpm* Panel function for mean smooths used with `bmmp`

panel.bmmpv* Panel function for variance smooths used with `bmmp`

postsamp.lm Samples from the posterior distribution of a linear model

postsamp.gam* Extract samples from Hastie and Tibshirani's `gibbs.gam` function (this is an S-PLUS function for sampling from the posterior of an additive model—see Hastie and Tibshirani 2000)

postsamp.binlog Extract binary logistic samples from BUGS/BOA output (BUGS is a stand-alone piece of software to perform Bayesian inference using Gibbs sampling—see Spiegelhalter, Thomas, and Best 1999; BOA is a set of functions for analyzing BUGS output in S-PLUS or R—see Smith 2001)

mmp Construct a marginal model plot for a fitted model, which can be a `lm`, `gam` or `glm` object

mmp.gam* Support function for `mmp` that works with `gam` objects

mmp.glm Support function for `mmp` that works with `glm` objects

mmp.lm Support function for `mmp` that works with `lm` objects

mmp.default Support function for `mmp` that does the final calculations and plotting

4   How to use the functions

Section 4.1 describes how to use the function `bmmp` which constructs Bayes marginal model plots. The only required arguments are a fitted model object of class inheriting from `lm`, `gam` or `glm`, and an object containing posterior samples. This latter object should have a component `yhat`, a matrix with *n* rows and *nsamp* columns, and, optionally, a component `vhat`, a vector with *nsamp* elements. Here, *n* is the number of cases fit by the model and *nsamp* is the number of posterior samples generated. Sections 4.2, 4.3 and 4.4 describe how to use functions that construct this posterior sample object for a variety of circumstances; the code for these functions could be used as templates for constructing posterior samples in other circumstances.

Section 4.5 describes how to use the function `mmp` which constructs marginal model plots. The only required argument is a fitted model object of class inheriting from `lm`, `gam` or `glm`. This function is really just a more limited version of the `bmmp` function, and as such is not a necessary part of the software. However it is included so that marginal model plots can be constructed in situations where posterior samples are not available, or a quick visualization of the fit of a model is wanted.

4.1 `bmmp`

Construct a Bayes marginal model plot

Description

`bmmp` is used to construct a Bayes marginal model plot for a fitted regression model, which can be a linear model, an additive model, or a generalized linear model.

Usage

```
bmmp(x, samp, h = NULL, default = T, random = NULL,
     sptry = NULL, spartry = NULL, sparams = NULL,
     spar = NULL, spdefault = F, incvar = T, mult = .15,
     included = NULL, usetrellis = T, smoother = "spline",
     bdm = T, printopts = 0)
```

Arguments

x	object of class inheriting from `lm`, `gam` or `glm`.
samp	object containing posterior samples; this should have a component `yhat`, a matrix with *n* rows and *nsamp* columns, and, optionally, a component `vhat`, a vector with *nsamp* elements. Here, *n* is the number of cases fit by the model and *nsamp* is the number of posterior samples generated.
h	an optional list of predictors or functions of predictors to be used as h functions for plotting on the horizontal axes of the plots (in addition to the default h if `default=T`, or instead of the default h if `default=F`).
default	a logical value indicating whether a plot for h = fitted values (for `lm` objects), h = additive fit (for `gam` objects), or h = linear fit (for `glm` objects) should be constructed.
random	an optional (integer valued) number indicating the number of "random direction" plots to construct.
sptry	an optional list of numbers to be used as smoothing parameters for the smooths displayed in the "smoothing parameter choice" plots; this should be no more than five numbers to avoid cluttering the plots; equivalent to `df` for "spline", `span` for "loess1" and "loess2", a multiple of `bandwidth` for "kernel" and `span` for "supsmu".
spartry	an optional list of alternative smoothing parameters to be tried when using method "spline", equivalent to `spar`.

sparams	an optional list of smoothing parameters to be used for the smooths in each of the plots; the number of elements of sparams must be equal to the number of elements of h (plus one if default=T, plus random if random is non-null); needs to be appropriate for the type of smoothing method (see sptry).
spar	an optional list of alternative smoothing parameters when using method "spline".
spdefault	a logical value indicating whether default values (chosen by the software) should be used for the smoothing parameters; cannot be set to T if sparams or spar is non-null.
incvar	a logical value indicating whether variance function smooths should be constructed.
mult	an optional number controlling the additional space allowed for on the vertical axes in the plots to allow all the smooths to be displayed; expressed as a percentage of the vertical axis range.
included	an optional vector specifying a subset of observations to be used in constructing the plots.
usetrellis	a logical value indicating whether Trellis graphics or traditional graphics should be used; T by default for S-PLUS, but F by default for R since Trellis graphics are not implemented in R.
smoother	the smoothing method to be used, by default "smooth" for cubic smoothing splines, but alternatively "loess1" for linear loess smoothing, "loess2" for quadratic loess smoothing, "kernel" for kernel smoothing or "supsmu" for Friedman's super smoother.
bdm	a logical value indicating whether the "Bayes discrepancy measure" (Pardoe and Cook, 2002) for each plot should be calculated and summarized.
printopts	an (integer valued) number indicating graphical settings to be used, by default "0" for display on a color monitor, but alternatively "1" for inverse display on a color monitor (useful for presentations that use a light foreground on a dark background), "2" for monochrome printing, or "3" for color printing.

Details

The two required arguments are the fitted model x and the posterior samples samp. The plots are implemented differently depending on whether x is a lm, gam or glm object, but this is all handled automatically. Other arguments control the number and type of plots constructed, how smoothing is implemented, and graphical features of the plots.

Value

bmmp constructs the appropriate number of Bayes marginal model plots and draws them on the current graphical device. If bdm=T then a summary of the Bayes discrepancy measure calculations is displayed.

Note

If Trellis graphics are being used (S-PLUS only), the device must be able to display such graphics: use `trellis.device()` before constructing the plots.

See Also

`mmp` for marginal model plots; `postsamp.lm` to create samples from linear models for use with `bmmp`; `postsamp.gam` to create samples from additive models for use with `bmmp`; `postsamp.binlog` to create samples from binary logistic models for use with `bmmp`.

Examples

The following examples use the "naphthalene" data that are described in Pardoe (2001). This dataset (`naphthalene.dat`) is available at:

    http://lcb1.uoregon.edu/ipardoe/research/bmmpsoft.htm

Download the dataset and move it to a subdirectory of your working directory called `data`. Then start S-PLUS (or R) and read in the data and fit a full second order *linear model* using the following:

```
naph <- read.table("data/naphthalene.dat", header = T,
                   row.names = NULL)
attach(naph)
fit <- lm(Yn ~ (AN+Btemp+Ctime)^2 + I(AN^2) + I(Btemp^2) +
          I(Ctime^2))
```

Next, obtain posterior samples from this model using prior $\propto$ variance^{-1}—for details regarding this prior see Section 4.2:

```
nsamp <- 100
samp <- postsamp.lm(fit, nsamp = nsamp)
```

Next, open a graphics device. In S-PLUS, Trellis graphics or traditional graphics can be used; in R, only traditional graphics are available. Enter one of the following commands:

```
trellis.device() # trellis graphics device (S-PLUS only)
motif() # traditional graphics device (S-PLUS UNIX/Linux)
graphsheet() # traditional graphics device (S-PLUS Windows)
x11() # traditional graphics device (R UNIX/Linux)
windows() # traditional graphics device (R Windows)
```

If you are using traditional graphics in S-PLUS, include the argument `usetrellis=F` in the commands below; in R, `usetrellis` is `F` by default.

The *default* plot (with horizontal axis equal to the fitted values from the model, and smoothing method equal to "spline") is obtained using the following (if you are using R, attach the `modreg` library first using `library(modreg)`):

```
bmmp(fit, samp = samp)
```

The first display you see prompts you for a smoothing parameter: select the number corresponding to the smooth which gives the best visual fit in *both* plots (that is, tracking clear patterns but not over-reacting to individual points). In this example, the smooth with smoothing parameter equal to three seems to provide the best compromise between over- and under-smoothing. After entering your choice for smoothing parameter, a marginal model plot with horizontal axis h equal to the fitted values from the model is displayed. The superimposed spline smooths are based on the smoothing parameter you chose previously.

Next, you are prompted to "construct Bayes marginal model plot(s) for mean" or to return to the command line. After entering "1", a Bayes marginal model plot for the mean function with horizontal axis h equal to the fitted values from the model is displayed. The superimposed spline smooths are again based on the smoothing parameter you chose previously. Then, you are prompted to "construct Bayes marginal model plot(s) for variance" or to return to the command line. After entering "1", a Bayes marginal model plot for the variance function with horizontal axis h equal to the fitted values from the model is displayed. Again, the superimposed spline smooths are based on the smoothing parameter you chose previously. For an explanation of how the plots are constructed and how they can be interpreted, see Pardoe (2001).

A summary of the Bayes discrepancy measure calculations is displayed in the command window for each Bayes marginal model plot. See Pardoe and Cook (2002) for background on the calculation and interpretation of this measure.

Multiple plots can be obtained by specifying additional quantities for the h functions plotted on the horizontal axes of the plots. For example, plots for $h =$ fitted values and $h = AN$ (one of the predictors in the model) result from:

```
bmmp(fit, samp = samp, h = AN)
```

More than one additional h function can be specified by using `cbind`, for example:

```
bmmp(fit, samp = samp, h = cbind(AN, Btemp))
```

The default plot can be excluded by setting `default=F`:

```
bmmp(fit, samp = samp, h = AN, default = F)
```

Additional *random* projections of the predictors in the model can be specified by setting `random` equal to the number of random projections desired, for example:

```
bmmp(fit, samp = samp, default = F, random = 2)
```

The random projections are calculated by projecting a random standard normal p-vector (where p is the number of predictors in the model) onto the column-space spanned by the (non-constant) columns of the model-matrix (centered by subtracting column means and rescaled to have unit variance). So, a random projection r is calculated as:

$$ r = \frac{1}{\|z\|} \widetilde{X} z $$

where $z \sim \mathrm{N}(0_p, I_p)$ and $\widetilde{X}$ is the centered-scaled model matrix (excluding the constant column for any intercept).

It is recommended that no more than six plots are constructed at any one time—any more than this and the plots can appear very small and become difficult to interpret. Attempting to construct more than six plots using traditional graphics may also result in unsatisfactory configurations of plots and poor labeling.

If none of the four smooths at the "smoothing parameter choice" stage provide a good visual fit in both plots, *alternative smoothing parameters* can be tried by setting sptry to be a vector of numbers, for example:

```
bmmp(fit, samp = samp, sptry = c(2, 3, 4, 5))
```

With the "spline" method, the smoothing parameters can be specified in a different way using the argument spar for the smooth.spline function. See the help files in S-PLUS and R for further information on spar, as the implementation differs slightly on the different platforms and some experimentation may be needed to obtain useful values to try. This argument can be used by setting spartry to be a vector of numbers, for example:

```
bmmp(fit, samp = samp, spartry = c(.04, .01, .003, .001))
      # S-PLUS
bmmp(fit, samp = samp, spartry = c(1.3, 1.2, 1.1, 1)) # R
```

If plots have been constructed previously and it is known which smoothing parameter values provide good visual fits, the "smoothing parameter choice" stage can be skipped by *specifying the smoothing parameter values* to use in the construction of the marginal model plots. Set sparams (or alternatively spar for smoothing splines) to be a vector of numbers as follows:

```
bmmp(fit, samp = samp, h = AN, sparams = c(3, 3))
bmmp(fit, samp = samp, h = AN, spar = c(.04, .06)) # S-PLUS
bmmp(fit, samp = samp, h = AN, spar = c(1.3, 1.1)) # R
```

Make sure that the length of the vector argument is equal to the number of elements of `h` (plus one if `default=T`, plus `random` if `random` is non-null).

One final option (which is not really recommended) is to use the *smoothing parameter default values* built into the software. These are tailored to the smoothing method but pay no heed to the actual dataset in use. This can lead to misleading plots if these default values would not have given a good visual fit in the "smoothing parameter choice" plots. It can provide a useful starting point for a more rigorous analysis however. To use the default smoothing parameter values, set `spdefault=T`:

```
bmmp(fit, samp = samp, spdefault = T)
```

If you are only interested in the mean function smooths, the *variance function smooths* and plot(s) can be suppressed by setting `incvar=F`:

```
bmmp(fit, samp = samp, sparams = 3, incvar = F)
```

Additional vertical space is used in the construction of the plot to allow all the smooths to be displayed. The amount of space is controlled by the argument `mult` (set by default to 0.15), and is expressed as a percentage of the vertical axis range. Increase this quantity if more space is needed on the plot to accommodate all of the smooths, for example:

```
bmmp(fit, samp = samp, sparams = 3, mult = .2)
```

Subsets of the data can be specified, if, for example, you wish to assess the fit of the model for only some of the cases. The `included` argument should be a vector specifying a subset of observations to be used in constructing the plots, for example:

```
bmmp(fit, samp = samp, h = AN, sparams = c(3, 3),
     included = names(AN[AN<2])) # S-PLUS
bmmp(fit, samp = samp, h = AN, sparams = c(3, 3),
     included = as.character(seq(along=AN))[AN<2]) # R
```

In S-PLUS, Trellis graphics are used by default, but traditional graphics can be used by setting `usetrellis=F` (remember to open a traditional graphics device first using `motif()` for UNIX/Linux and `graphsheet()` for Windows):

```
bmmp(fit, samp = samp, usetrellis = F)
```

Note that the current graphics device must be of the appropriate type since Trellis and traditional graphics devices do not mix well. In R, traditional graphics are used by default, and setting `usetrellis=T` results in an error message.

Different smoothing methods can be tried if the default "spline" method is unsatisfactory for the dataset being analyzed. Currently, `"loess1"` specifies linear loess smoothing,

"loess2" quadratic loess smoothing, "kernel" kernel smoothing and "supsmu" Friedman's super smoother. Other smoothers could be added by writing code based on the examples provided by these other methods—the details vary slightly with each of them because of the varying outputs from the smoothers. The following inputs provide examples of the use of the smoother argument:

```
bmmp(fit, samp = samp, smoother = "loess1")
bmmp(fit, samp = samp, smoother = "loess1",
     h = cbind(AN, Btemp, Ctime),
     sparams = c(.7, .8, .8, .8))
bmmp(fit, samp = samp, smoother = "loess2")
bmmp(fit, samp = samp, smoother = "loess2",
     h = cbind(AN, Btemp, Ctime),
     sparams = c(1, 1, 1, .9))
bmmp(fit, samp = samp, smoother = "kernel")
bmmp(fit, samp = samp, smoother = "kernel",
     h = cbind(AN, Btemp, Ctime),
     sparams = c(.2, .4, .3, .3))
bmmp(fit, samp = samp, smoother = "supsmu", incvar = F)
bmmp(fit, samp = samp, smoother = "supsmu", incvar = F,
     h = cbind(AN, Btemp, Ctime),
     sparams = c(.5, .6, .6, .4))
```

Note that variance function smooths are not available with "supsmu" (because of the restricted output available for this smoothing method).

As an example of a different analysis of this dataset, fit an *additive model* using the following (S-PLUS for UNIX/Linux only, since Hastie and Tibshirani's gibbs.gam function is currently only available for this platform):

```
x <- .397*AN + .445*Ctime + .802*Btemp
fit <- gam(Yn ~ s(x, df = 4))
```

Hastie and Tibshirani (1990) describe a Bayesian characterization of additive models based on partially improper normal process priors for each of the smooth functions of the predictors. This allows posterior samples to be obtained for this model—see Section 4.3 for a description of how posterior samples can be obtained using Hastie and Tibshirani's gibbs.gam function (Hastie and Tibshirani, 2000). The single required argument for the postsamp.gam function is the object output from the gibbs.gam function. An example of this object (ggnaph) is available for download at:

<div align="center">http://lcb1.uregon.edu/ipardoe/research/bmmpsoft.htm</div>

To use this object without having to use gibbs.gam, download and unzip the file, and then move it to a subdirectory of your working directory called data. Read it into S-PLUS for UNIX/Linux using source:

<div align="center">10</div>

```
source("data/ggnaph")
```

Extract the posterior samples using:

```
samp <- postsamp.gam(ggnaph)
# ggnaph is the object output from gibbs.gam()
```

All the examples above should work for this model also, for example:

```
bmmp(fit, samp = samp, h = AN)
bmmp(fit, samp = samp, default = F, random = 1)
     # adds no information when p = 1
bmmp(fit, samp = samp, h = cbind(AN, Btemp, Ctime),
     sparams = c(4, 3, 3, 3))
bmmp(fit, samp = samp, h = AN, sparams = c(4, 3),
     included = names(AN[AN<2]))
bmmp(fit, samp = samp, smoother = "loess1")
bmmp(fit, samp = samp, smoother = "loess1",
     h = cbind(AN, Btemp, Ctime),
     sparams = c(.8, .8, .8, .8))
bmmp(fit, samp = samp, smoother = "loess2")
bmmp(fit, samp = samp, smoother = "loess2",
     h = cbind(AN, Btemp, Ctime),
     sparams = c(1, 1, 1, 1))
bmmp(fit, samp = samp, smoother = "kernel")
bmmp(fit, samp = samp, smoother = "kernel",
     h = cbind(AN, Btemp, Ctime),
     sparams = c(.2, .4, .3, .3))
bmmp(fit, samp = samp, smoother = "supsmu", incvar = F)
bmmp(fit, samp = samp, smoother = "supsmu", incvar = F,
     h = cbind(AN, Btemp, Ctime),
     sparams = c(.6, .6, .6, .6))
```

Note that the *default* plot for additive models has horizontal axis equal to the additive fit from the model. Finally, clean up:

```
detach("naph")
rm(fit, naph, nsamp, samp, x, ggnaph) # S-PLUS UNIX/Linux
rm(fit, naph, nsamp, samp) # S-PLUS Windows, R
```

The following examples use the "Wisconsin Breast Cancer" data that are described in Pardoe and Cook (2002). This dataset (`WBCD.dat`) is available at:

    http://lcb1.uoregon.edu/ipardoe/research/bmmpsoft.htm

Download the dataset and move it to a subdirectory of your working directory called `data`. Then, if necessary, start S-PLUS (or R) and read in the data and fit a *binary logistic model* using the following:

11

```
wbcd <- read.table("data/WBCD.dat", header = T,
                      row.names = NULL)
attach(wbcd)
fit <- glm(Class1 ~ Adhes + BNucl + Chrom + NNucl + Thick,
             family = binomial)
```

Next, obtain posterior samples from this model based on a "vague" multivariate normal prior—see Section 4.4 for a description of how posterior samples can be obtained using **BUGS** and **BOA**. One of the required arguments for the `postsamp.binlog` function is the object output from **BOA**. An example of this object (`boawbcd`) is available for download at:

> `http://lcb1.uoregon.edu/ipardoe/research/bmmpsoft.htm`

To use this object without having to use **BUGS** or **BOA**, download and unzip the file, and then move it to a subdirectory of your working directory called `data`. Read it into **S-PLUS** using `data.restore` or into **R** using `load`:

```
data.restore("data/boawbcd") # S-PLUS
load("data/boawbcd") # R
```

Extract the posterior samples using:

```
nsamp <- 100
samp <- postsamp.binlog(fit, boaobject = boawbcd,
                              nsamp = nsamp)
# boawbcd is the object output from BOA
```

All the examples above should work for this model also, for example:

```
bmmp(fit, samp = samp, h = Mitos, sptry = c(4, 8, 12, 16),
     incvar = F)
bmmp(fit, samp = samp, default = F, random = 2, incvar = F)
bmmp(fit, samp = samp, h = cbind(Mitos, Adhes, BNucl),
     sparams = c(12, 4, 4, 4), incvar = F)
bmmp(fit, samp = samp, h = Mitos, sparams = c(12, 4),
     incvar = F, included = names(Mitos[Mitos<10])) # S-PLUS
bmmp(fit, samp = samp, h = Mitos, sparams = c(12, 4),
     included = as.character(seq(along=Mitos))[Mitos<10],
     incvar = F) # R
bmmp(fit, samp = samp, smoother = "loess1", incvar = F,
     sptry = c(.4, .3, .2, .1))
bmmp(fit, samp = samp, smoother = "loess1", incvar = F,
     h = cbind(Mitos, Adhes, BNucl),
     sparams = c(.1, .95, .7, .7))
```

```
bmmp(fit, samp = samp, smoother = "loess2", incvar = F,
     sptry = c(.4, .3, .2, .1))
bmmp(fit, samp = samp, smoother = "loess2", incvar = F,
     h = cbind(Mitos, Adhes, BNucl),
     sparams = c(.3, .9, .85, .9))
bmmp(fit, samp = samp, smoother = "kernel", incvar = F,
     sptry = c(.4, .3, .2, .1))
bmmp(fit, samp = samp, smoother = "kernel", incvar = F,
     h = cbind(Mitos, Adhes, BNucl),
     sparams = c(.1, .3, .3, .3))
bmmp(fit, samp = samp, smoother = "supsmu", incvar = F,
     sptry = c(.4, .3, .2, .1))
bmmp(fit, samp = samp, smoother = "supsmu", incvar = F,
     h = cbind(Mitos, Adhes, BNucl),
     sparams = c(.1, .1, .1, .1))
```

Note that the *default* plot for generalized linear models has horizontal axis equal to the linear fit from the model. Finally, clean up:

```
detach("wbcd")
rm(fit, wbcd, boawbcd, nsamp, samp)
```

4.2 `postsamp.lm`

Samples from the posterior distribution of a linear model

Description

`postsamp.lm` is used to construct an object containing posterior samples for a normal linear regression model using the usual non-informative prior $\propto$ variance^{-1}. The output can be used as an input to function `bmmp`.

Usage

```
postsamp.lm(lmobject, nsamp = 100)
```

Arguments

lmobject   object of class inheriting from `lm`.
nsamp   an optional (integer-valued) number indicating the number of posterior
            samples wanted.

Details

This function generates expected response values and variances based on posterior sampling from fitted model `lmobject` using prior $\propto$ variance^{-1}.

Value

`postsamp.lm` constructs an object with components `yhat` and `vhat`. Component `yhat` is a matrix with *n* rows and `nsamp` columns containing `nsamp` replicate datasets of *n* response values. Here, *n* is the number of cases fit by the model. Component `vhat` is a vector containing `nsamp` variance values, one for each replicated dataset.

Note

Posterior sampling is based on the usual non-informative prior $\propto$ variance^{-1}. The code for this function could be used as a template to sample for other priors.

See Also

`bmmp` for Bayes marginal model plots; `postsamp.gam` to create samples from additive models for use with `bmmp`; `postsamp.binlog` to create samples from binary logistic models for use with `bmmp`.

Examples

The following example again uses the "naphthalene" data. Read in the data and fit a full second order *linear model* using the following:

```
naph <- read.table("data/naphthalene.dat", header = T,
                   row.names = NULL)
attach(naph)
fit <- lm(Yn ~ (AN+Btemp+Ctime)^2 + I(AN^2) + I(Btemp^2) +
          I(Ctime^2))
```

The normal linear regression model can be written

$$y_i | \boldsymbol{x}_i = \mathrm{E}(y | \boldsymbol{x}_i) + e_i / \sqrt{w_i}, \quad i = 1, \ldots, n$$

where $\mathrm{E}(y|\boldsymbol{x}_i) = \boldsymbol{\beta}^T \boldsymbol{x}_i$, $\boldsymbol{\beta}$ is a $p \times 1$ vector of unknown parameters, $\boldsymbol{x}_i$ is the $p \times 1$ vector of predictor values for the i-th observation, the errors e_i are normally distributed with mean 0 and variance σ^2, and the weights $w_i > 0$ are known, positive numbers. An intercept term can be included within this framework by setting one of the predictors equal to a constant. Defining $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T$, $\boldsymbol{\theta} = (\boldsymbol{\beta}^T, \sigma^2)^T$, and $\boldsymbol{W} = \mathrm{diag}(w_i)$, then (suppressing the notation for conditioning on M for clarity)

$$\boldsymbol{y} | (\boldsymbol{X}, \boldsymbol{\theta}) \sim \mathrm{N}\left(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2 \boldsymbol{W}^{-1}\right)$$

The usual non-informative prior for the normal linear regression model is $\pi(\boldsymbol{\theta}) \propto \sigma^{-2}$. Consider constructing BMMPs in this situation. Since the prior is improper, only Rubin's approach is appropriate. Sampling from the posterior is straightforward since

$$\pi(\boldsymbol{\beta}, \sigma^2 | \boldsymbol{X}, \boldsymbol{y}_d) = \pi(\boldsymbol{\beta} | \boldsymbol{X}, \boldsymbol{y}_d, \sigma^2) \, \pi(\sigma^2 | \boldsymbol{X}, \boldsymbol{y}_d)$$

14

In particular, draw a value of σ^2 from

$$\sigma^2 | (\boldsymbol{X}, \boldsymbol{y}_d) \sim \text{RSS } \chi_{n-p}^{-2}$$

where RSS χ_{n-p}^{-2} is the usual weighted residual sum of squares divided by a chi-squared random variable with $n - p$ degrees of freedom. Then, holding σ^2 fixed, draw a value of $\boldsymbol{\beta}$ from

$$\boldsymbol{\beta} | (\boldsymbol{X}, \boldsymbol{y}_d, \sigma^2) \sim \text{N}\left(\widehat{\boldsymbol{\beta}}, \sigma^2 (\boldsymbol{X}^T \boldsymbol{W} \boldsymbol{X})^{-1}\right)$$

where $\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T \boldsymbol{W} \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{W} \boldsymbol{y}_d$ is the usual weighted least squares estimate for $\boldsymbol{\beta}$. Sampling $\boldsymbol{\beta}$ is very fast using the QR decomposition of $\boldsymbol{W}^{1/2} \boldsymbol{X}$. In particular, note that

$$\widehat{\boldsymbol{\beta}} + \sigma \boldsymbol{R}^{-1} \boldsymbol{z} \sim \text{N}\left(\widehat{\boldsymbol{\beta}}, \sigma^2 (\boldsymbol{X}^T \boldsymbol{W} \boldsymbol{X})^{-1}\right)$$

where $\boldsymbol{R}$ is the upper triangular matrix from the QR decomposition of $\boldsymbol{W}^{1/2} \boldsymbol{X}$ which has the property $\boldsymbol{R}^T \boldsymbol{R} = \boldsymbol{X}^T \boldsymbol{W} \boldsymbol{X}$, and $\boldsymbol{z} \sim \text{N}(\boldsymbol{0}_n, \boldsymbol{I}_n)$, where $\boldsymbol{0}_n$ is an $n \times 1$ vector of zeros, and $\boldsymbol{I}_n$ is the $n \times n$ identity matrix. $\boldsymbol{R}$ is often a part of the output from linear model fitting in statistical software packages.

Constructing a *BMMP for the mean* in direction h requires model-free and model-based estimates of the mean function with respect to h. To obtain the model-free estimate $\widehat{\text{E}}_{\text{F}}(y|h)$, smooth the data $\{y_i\}$ on $\{h_i\}$. To obtain the model-based estimates $\widehat{\text{E}}_{\text{M}_{\boldsymbol{\theta}_t}}(y|h)$, smooth the fitted-values based on the posterior samples $\{\text{E}_{\text{M}_{\boldsymbol{\theta}_t}}(y|\boldsymbol{x}_i)\}$ on $\{h_i\}$. The fitted values corresponding to the posterior samples $\boldsymbol{\beta}_t$ are

$$\text{E}_{\text{M}_{\boldsymbol{\theta}_t}}(y|\boldsymbol{x}_i) = \boldsymbol{\beta}_t^T \boldsymbol{x}_i, \quad i = 1, \ldots, n; \; t = 1, \ldots, m$$

Constructing a *BMMP for the variance* in direction h requires model-free and model-based estimates of the variance function with respect to h. To obtain the model-free estimate $\widehat{\text{Var}}_{\text{F}}(y|h)$, smooth $\{(y_i - \widehat{\text{E}}_{\text{F}}(y|h_i))^2\}$ on $\{h_i\}$. To obtain the model-based estimates $\widehat{\text{Var}}_{\text{M}_{\boldsymbol{\theta}_t}}(y|h)$, add smooths of $\{\text{Var}_{\text{M}_{\boldsymbol{\theta}_t}}(y|\boldsymbol{x}_i)\}$ on $\{h_i\}$ to smooths of $\{(\text{E}_{\text{M}_{\boldsymbol{\theta}_t}}(y|\boldsymbol{x}_i) - \widehat{\text{E}}_{\text{M}_{\boldsymbol{\theta}_t}}(y|h_i))^2\}$ on $\{h_i\}$. The variance values used to obtain $\text{Var}_{\text{M}_{\boldsymbol{\theta}_t}}(y|\boldsymbol{x}_i)$ are the σ^2 draws divided by the weights w_i. The following generates these posterior samples from the model:

```
nsamp <- 100
samp <- postsamp.lm(fit, nsamp = nsamp)
```

See Section 4.1 for how to use these samples in the construction of Bayes marginal model plots.

4.3 `postsamp.gam`

Extract samples from Hastie and Tibshirani's `gibbs.gam` function

Description

`postsamp.gam` is used to construct an object containing posterior samples for an additive regression model using output from Hastie and Tibshirani's `gibbs.gam` function (Hastie and Tibshirani, 2000). The output can be used as an input to function `bmmp`.

Usage

```
postsamp.gam(htobject)
```

Arguments

`htobject`   object output from Hastie and Tibshirani's `gibbs.gam` function

Details

This function generates expected response values based on posterior sampling from an additive model obtained using `gibbs.gam`.

Value

`postsamp.gam` constructs an object with component `yhat`, a matrix with *n* rows and `nsamp` columns containing `nsamp` replicate datasets of *n* response values. Here, *n* is the number of cases fit by the model.

Note

The actual posterior sampling is carried out using Hastie and Tibshirani's `gibbs.gam` function; output from that function is then used as an input to `postsamp.gam` to construct the appropriate input for `bmmp`.

See Also

`bmmp` for Bayes marginal model plots; `postsamp.lm` to create samples from normal linear models for use with `bmmp`; `postsamp.binlog` to create samples from binary logistic models for use with `bmmp`.

Examples

The following example again uses the "naphthalene" data. Read in the data and fit an *additive model* using the following (S-PLUS for UNIX/Linux only, since Hastie and Tibshirani's `gibbs.gam` function is currently only available for this platform):

```
naph <- read.table("data/naphthalene.dat", header = T,
                   row.names = NULL)
attach(naph)
```

```
x <- .397*AN + .445*Ctime + .802*Btemp
fit <- gam(Yn ~ s(x, df = 4))
```

The additive regression model can be written

$$y_i | \boldsymbol{x}_i = \alpha + \sum_{j=1}^{p} f_j(x_{ij}) + e_i, \quad i = 1, \ldots, n$$

where f_j is a "smooth" function for the j-th predictor, x_{ij} is the i-th observation of the j-th predictor, and the errors e_i are normally distributed with mean 0 and variance σ^2. Defining $\boldsymbol{\theta} = (\alpha, f_1, \ldots, f_p, \sigma^2)^T$, then

$$\boldsymbol{y}|(\boldsymbol{X}, \boldsymbol{\theta}) = \alpha \boldsymbol{J}_n + \sum_{j=1}^{p} \boldsymbol{f}_j + \boldsymbol{e} \tag{1}$$

where $\boldsymbol{J}_n$ is an $n \times 1$ vector of ones, $\boldsymbol{f}_j = (f_j(x_{1j}), \ldots, f_j(x_{nj}))^T$, and $\boldsymbol{e}$ is an $n \times 1$ vector of errors.

Hastie and Tibshirani (1990) describe a Bayesian characterization of (1) based on partially improper normal process priors for each $\boldsymbol{f}_j$

$$\boldsymbol{f}_j \sim \mathrm{N}(\boldsymbol{0}_n, \tau_j^2 \boldsymbol{K}_j^-)$$

where $\boldsymbol{K}_j^-$ is a generalized inverse of a matrix $\boldsymbol{K}_j$ which is related to the construction of the estimate of $\boldsymbol{f}_j$. For example, when $\boldsymbol{f}_j$ is estimated using a symmetric smoother matrix $\boldsymbol{S}_j$ with smoothing parameter λ_j, then $\tau_j^2 = \sigma^2/\lambda_j$, and $\boldsymbol{K}_j^- = \lambda_j (\boldsymbol{S}_j^- - \boldsymbol{I}_n)^-$, where $\boldsymbol{I}_n$ is the $n \times n$ identity matrix.

Consider constructing BMMPs in this situation. Rubin's approach is appropriate here if most of the information about $\boldsymbol{\theta}$ is going to come from the data rather than the prior. Also, the prior for $\boldsymbol{\theta}$ can be improper if any of the $\boldsymbol{f}_j$ correspond to fixed linear effects, thus necessitating Rubin's approach. Hastie and Tibshirani (2000) derive a way to sample from the posterior of $\boldsymbol{\theta}$ using a stochastic generalization of the backfitting algorithm based on *Gibbs sampling*. The Gibbs sampler is described in more detail in Section 4.4. In particular, $\boldsymbol{f}_j$ has posterior

$$\boldsymbol{f}_j|(\boldsymbol{X}, \boldsymbol{y}_d) \sim \mathrm{N}(\boldsymbol{S}_j \, \boldsymbol{y}_d, \sigma^2 \boldsymbol{S}_j)$$

Then, with σ^2 and each τ_j^2 held fixed, posterior samples are generated by adding noise $\sigma \boldsymbol{S}_j^{1/2} \boldsymbol{z}$, where $\boldsymbol{z}$ is an $n \times 1$ vector of standard normal random variables, to the partial residual smooths in the backfitting algorithm. If σ^2 and each τ_j^2 are not held fixed, conjugate inverse gamma priors lead to inverse gamma conditional sampling steps for σ^2 and each τ_j^2 in the algorithm. Hastie and Tibshirani have developed S-PLUS functions to carry out their sampling procedure.

Constructing a *BMMP for the mean* in direction h requires model-free and model-based estimates of the mean function with respect to h. To obtain the model-free estimate

$\widehat{\mathrm{E}}_{\mathrm{F}}(y|h)$, smooth the data $\{y_i\}$ on $\{h_i\}$. To obtain the model-based estimates $\widehat{\mathrm{E}}_{\mathrm{M}_{\theta_t}}(y|h)$, smooth the fitted-values based on the posterior samples $\{\mathrm{E}_{\mathrm{M}_{\theta_t}}(y|\boldsymbol{x}_i)\}$ on $\{h_i\}$. The fitted values corresponding to posterior samples $(\alpha_t, \boldsymbol{f}_{1t}, \ldots, \boldsymbol{f}_{pt})$ are

$$\mathrm{E}_{\mathrm{M}_{\theta_t}}(y|\boldsymbol{x}_i) = \alpha_t + f_{1t}(x_{i1}) + \cdots + f_{pt}(x_{ip}), \quad i = 1, \ldots, n; \ t = 1, \ldots, m$$

See Section 6 for an outline of how to use `gibbs.gam` to generate these posterior samples from the model for this example (Hastie and Tibshirani 2000 provide details on obtaining and using `gibbs.gam`). The required argument for the `postsamp.gam` function is the object output from the `gibbs.gam` function. An example of this object (`ggnaph`) is available for download at:

> `http://lcb1.uoregon.edu/ipardoe/research/bmmpsoft.htm`

To use this object without having to use `gibbs.gam`, download and unzip the file, and then move it to a subdirectory of your working directory called `data`. Read it into S-PLUS for UNIX/Linux using `source`:

```
source("data/ggnaph")
```

Extract the posterior samples using:

```
samp <- postsamp.gam(ggnaph)
# ggnaph is the object output from gibbs.gam()
```

See Section 4.1 for how to use these samples in the construction of Bayes marginal model plots.

4.4 `postsamp.binlog`

Extract binary logistic samples from BUGS/BOA output

Description

`postsamp.binlog` is used to construct an object containing posterior samples for a binary logistic regression model using BUGS/BOA output. The output can be used as an input to function `bmmp`.

Usage

```
postsamp.binlog(glmobject, boaobject, nsamp = 100)
```

Arguments

`glmobject`	object of class inheriting from `glm`.
`boaobject`	object output from BOA.
`nsamp`	an optional (integer-valued) number indicating the number of posterior samples wanted.

18

Details

This function generates expected response values based on posterior sampling from a binary logistic model obtained using BUGS and BOA.

Value

`postsamp.binlog` constructs an object with component `yhat`, a matrix with n rows and `nsamp` columns containing `nsamp` replicate datasets of n response values. Here, n is the number of cases fit by the model.

Note

Posterior sampling is carried out in BUGS, the results of which are output to S-PLUS or R via BOA. The code for this function could be used as a template to extract posterior samples for other models for which posterior sampling has been carried out in BUGS.

See Also

`bmmp` for Bayes marginal model plots; `postsamp.lm` to create samples from normal linear models for use with `bmmp`; `postsamp.gam` to create samples from additive models for use with `bmmp`.

Examples

The following example again uses the "Wisconsin Breast Cancer" data. Read in the data and fit a *binary logistic model* using the following:

```
wbcd <- read.table("data/WBCD.dat", header = T,
                    row.names = NULL)
attach(wbcd)
fit <- glm(Class1 ~ Adhes + BNucl + Chrom + NNucl + Thick,
           family = binomial)
```

The binary logistic regression model, a much-used example of a generalized linear model (McCullagh and Nelder, 1989), can be written

$$y_i|(\boldsymbol{x}_i, p_i) \sim \text{Bernoulli}(p_i)$$

where

$$p_i = \Pr(y = 1|\boldsymbol{x}_i) = \text{E}(y|\boldsymbol{x}_i)$$

$$\text{logit}(p_i) = \log\left(\frac{p_i}{1 - p_i}\right) = \boldsymbol{\beta}^T\boldsymbol{x}_i$$

One possible prior for this example is

$$\boldsymbol{\beta} \sim \text{N}(\boldsymbol{0}_p, k\boldsymbol{I}_p) \tag{2}$$

where k can be set to reflect the degree of prior uncertainty for any particular dataset. If k is set so that the prior is very vague, Rubin's approach is appropriate, and samples from the posterior are needed. It is not possible to sample directly from the posterior, so instead Markov chain simulation can be used to simulate a random walk in the parameter space which converges to a stationary distribution that is the desired joint distribution (as long as the Markov chain converges appropriately).

A particular Markov chain algorithm that is useful in many multidimensional problems such as this is *alternating conditional sampling*, also called the Gibbs sampler. Each iteration of the Gibbs sampler cycles through the components of $\boldsymbol{\beta}$, drawing each component conditional on the values of all the others. These (univariate) conditional posterior distributions can often be sampled from relatively easily. Casella and George (1992) provide a useful introduction to the use of the Gibbs sampler in statistics. In particular, posterior samples can be obtained using "Bayesian inference Using Gibbs Sampling" (BUGS): software developed by the biostatistics department at Cambridge University in England (Spiegelhalter, Thomas, and Best, 1999).

Checking convergence in Markov chain sampling is very important, and software is available from various sources to assist in this task. One particularly easy-to-use piece of software that works well with BUGS output is "Bayesian Output Analysis" (BOA)—see Smith (2001).

Constructing a *BMMP for the mean* in direction h requires model-free and model-based estimates of the mean function with respect to h. To obtain the model-free estimate $\widehat{\mathrm{E}}_{\mathrm{F}}(y|h)$, smooth the data $\{y_i\}$ on $\{h_i\}$. To obtain the model-based estimates $\widehat{\mathrm{E}}_{\mathrm{M}_{\boldsymbol{\theta}_t}}(y|h)$, smooth the fitted-values based on the posterior samples $\{\mathrm{E}_{\mathrm{M}_{\boldsymbol{\theta}_t}}(y|\boldsymbol{x}_i)\}$ on $\{h_i\}$. Here, $\boldsymbol{\theta} \equiv \boldsymbol{\beta}$, and the fitted values corresponding to posterior samples $\boldsymbol{\beta}_t$ are

$$\mathrm{E}_{\mathrm{M}_{\boldsymbol{\theta}_t}}(y|\boldsymbol{x}_i) = p_{it} = \frac{1}{1 + \exp(-\boldsymbol{\beta}_t^T \boldsymbol{x}_i)}, \quad i = 1, \ldots, n;\ t = 1, \ldots, m$$

See Section 7 for an outline of how to use BUGS and BOA to generate these posterior samples from the model for this example. One of the required arguments for the `postsamp.binlog` function is the object output from BOA. An example of this object (`boawbcd`) is available for download at:

`http://lcb1.uoregon.edu/ipardoe/research/bmmpsoft.htm`

To use this object without having to use BUGS or BOA, download and unzip the file, and then move it to a subdirectory of your working directory called `data`. Read it into S-PLUS using `data.restore` or into R using `load`:

```
data.restore("data/boawbcd") # S-PLUS
load("data/boawbcd") # R
```

Extract the posterior samples using:

```
nsamp <- 100
```

```
samp <- postsamp.binlog(fit, boaobject = boawbcd,
                               nsamp = nsamp)
# boawbcd is the object output from BOA
```

See Section 4.1 for how to use the posterior samples in the construction of Bayes marginal model plots.

4.5 `mmp`

Construct a marginal model plot

Description

`mmp` is used to construct a marginal model plot for a fitted regression model, which can be an additive model, a generalized linear model or a linear model.

Usage

```
mmp(x, h = NULL, default = T, random = NULL, sptry = NULL,
    spartry = NULL, sparams = NULL, spar = NULL,
    spdefault = F, adjvar = F, incvar = T, mult = .05,
    included = NULL, usetrellis = T, smoother = "spline")
```

Arguments

x	object of class inheriting from `lm`, `gam` or `glm`.
h	an optional list of predictors or functions of predictors to be used as h functions for plotting on the horizontal axes of the plots (in addition to the default h if `default=T`, or instead of the default h if `default=F`).
default	a logical value indicating whether a plot for $h =$ fitted values (for `lm` objects), $h =$ additive fit (for `gam` objects), or $h =$ linear fit (for `glm` objects) should be constructed.
random	an optional (integer valued) number indicating the number of "random direction" plots to construct.
sptry	an optional list of numbers to be used as smoothing parameters for the smooths displayed in the "smoothing parameter choice" plots; should be no more than five numbers to avoid cluttering the plots; equivalent to `df` for "spline", `span` for "loess1" and "loess2", a multiple of `bandwidth` for "kernel" and `span` for "supsmu".
spartry	an optional list of alternative smoothing parameters to be tried when using method "spline", equivalent to `spar`.
sparams	an optional list of smoothing parameters to be used for the smooths in each of the plots; the number of elements of `sparams` must be equal to the number of elements of `h` (plus one if `default=T`, plus `random` if `random` is non-null); needs to be appropriate for the type of smoothing method (see `sptry`).

21

spar	an optional list of alternative smoothing parameters when using method "spline".
spdefault	a logical value indicating whether default values (chosen by the software) should be used for the smoothing parameters; cannot be set to T if sparams or spar is non-null.
adjvar	a logical value indicating whether variance calculation bias adjustment as in Ruppert, Wand, Holst, and Hössjer (1997) should be used.
incvar	a logical value indicating whether variance function smooths should be constructed.
mult	an optional number controlling the additional space allowed for on the vertical axes in the plots to allow all the smooths to be displayed; expressed as a percentage of the vertical axis range.
included	an optional vector specifying a subset of observations to be used in constructing the plots.
usetrellis	a logical value indicating whether Trellis graphics or traditional graphics should be used; T by default for S-PLUS, but F by default for R since Trellis graphics are not implemented in R.
smoother	the smoothing method to be used, by default "smooth" for cubic smoothing splines, but alternatively "loess1" for linear loess smoothing, "loess2" for quadratic loess smoothing, "kernel" for kernel smoothing or "supsmu" for Friedman's super smoother.

Details

The only required argument is the fitted model x. The plots are implemented differently depending on whether x is a lm, gam or glm object, but this is all handled automatically. Other arguments control the number and type of plots constructed, how smoothing is implemented, and graphical features of the plots.

Value

mmp constructs the appropriate number of marginal model plots and draws them on the current graphical device.

Note

If Trellis graphics are being used (S-PLUS only), the device must be able to display such graphics: use trellis.device() before constructing the plots.

See Also

bmmp for Bayes marginal model plots; postsamp.lm to create samples from linear models for use with bmmp; postsamp.gam to create samples from additive models for use with bmmp; postsamp.binlog to create samples from binary logistic models for use with bmmp.

Examples

This section is an abbreviated version of the comparable section for `bmmp` above—for further details see that section. The following examples again use the "naphthalene" data:

```
naph <- read.table("data/naphthalene.dat", header = T,
                   row.names = NULL)
attach(naph)
fit <- lm(Yn ~ (AN+Btemp+Ctime)^2 + I(AN^2) + I(Btemp^2) +
          I(Ctime^2))
```

Next, open a graphics device:

```
trellis.device() # trellis graphics device (S-PLUS only)
motif() # traditional graphics device (S-PLUS UNIX/Linux)
graphsheet() # traditional graphics device (S-PLUS Windows)
x11() # traditional graphics device (R UNIX/Linux)
windows() # traditional graphics device (R Windows)
```

If you are using traditional graphics in S-PLUS, include the argument `usetrellis=F` in the commands below; in R, `usetrellis` is `F` by default.

The *default* plot (with horizontal axis equal to the fitted values from the model, and smoothing method equal to "spline") is obtained using the following (if you are using R, attach the `modreg` library first using `library(modreg)`):

```
mmp(fit)
```

Multiple plots can be obtained by specifying additional quantities for the h functions:

```
mmp(fit, h = AN)
mmp(fit, h = cbind(AN, Btemp))
```

The default plot can be excluded by setting `default=F`:

```
mmp(fit, h = AN, default = F)
```

Additional *random* projections of the predictors in the model can be specified:

```
mmp(fit, default = F, random = 2)
```

Alternative smoothing parameters can be tried by setting `sptry` or `spartry` ("spline" only):

```
mmp(fit, sptry = c(2, 3, 4, 5))
mmp(fit, spartry = c(.04, .01, .003, .001)) # S-PLUS
mmp(fit, spartry = c(1.3, 1.2, 1.1, 1)) # R
```

The "smoothing parameter choice" stage can be skipped by setting `sparams` (or alternatively `spar` for smoothing splines) to be a vector of numbers as follows:

```
mmp(fit, h = AN, sparams = c(3, 3))
mmp(fit, h = AN, spar = c(.04, .06)) # S-PLUS
mmp(fit, h = AN, spar = c(1.3, 1.1)) # R
```

To use the default smoothing parameter values, set `spdefault=T`:

```
mmp(fit, spdefault = T)
```

Set `adjvar=T` to incorporate a *variance calculation bias adjustment* as in Ruppert et al. (1997). Note that this is currently only implemented for the "spline" method for `mmp` (i.e. not for `bmmp` or for other smoothing methods), and can be very slow:

```
mmp(fit, sparams = 3, adjvar = T)
```

The *variance function smooths* can be suppressed by setting `incvar=F`:

```
mmp(fit, sparams = 3, incvar = F)
```

Additional vertical space can be allocated in the plots using `mult` (set by default to 0.05):

```
mmp(fit, sparams = 3, mult = .1)
```

The `included` argument can specify a subset of observations to be used in constructing the plots, for example:

```
mmp(fit, h = AN, sparams = c(3, 3),
    included = names(AN[AN<2])) # S-PLUS
mmp(fit, h = AN, sparams = c(3, 3),
    included = as.character(seq(along=AN))[AN<2]) # R
```

Traditional graphics can be used by setting `usetrellis=F` (remember to open a traditional graphics device first using `motif()` for UNIX/Linux and `graphsheet()` for Windows):

```
mmp(fit, usetrellis = F)
```

Different smoothing methods can be tried if the default "spline" method is unsatisfactory for the dataset being analyzed:

24

```
mmp(fit, smoother = "loess1")
mmp(fit, smoother = "loess1", h = cbind(AN, Btemp, Ctime),
    sparams = c(.7, .8, .8, .8))
mmp(fit, smoother = "loess2")
mmp(fit, smoother = "loess2", h = cbind(AN, Btemp, Ctime),
    sparams = c(1, 1, 1, .9))
mmp(fit, smoother = "kernel")
mmp(fit, smoother = "kernel", h = cbind(AN, Btemp, Ctime),
    sparams = c(.2, .4, .3, .3))
mmp(fit, smoother = "supsmu", incvar = F)
mmp(fit, smoother = "supsmu", h = cbind(AN, Btemp, Ctime),
    sparams = c(.5, .6, .6, .4), incvar = F)
```

As an example of a different analysis of this dataset, fit an *additive model* using the following (S-PLUS only, since gam is currently not available in R):

```
x <- .397*AN + .445*Ctime + .802*Btemp
fit <- gam(Yn ~ s(x, df = 4))
```

All the examples above should work for this model also, for example:

```
mmp(fit, h = AN)
mmp(fit, h = cbind(AN, Btemp, Ctime),
    sparams = c(4, 3, 3, 3))
mmp(fit, h = AN, sparams = c(4, 3),
    included = names(AN[AN<2]))
mmp(fit, smoother = "loess1")
mmp(fit, smoother = "loess1", h = cbind(AN, Btemp, Ctime),
    sparams = c(.8, .8, .8, .8))
mmp(fit, smoother = "loess2")
mmp(fit, smoother = "loess2", h = cbind(AN, Btemp, Ctime),
    sparams = c(1, 1, 1, 1))
mmp(fit, smoother = "kernel")
mmp(fit, smoother = "kernel", h = cbind(AN, Btemp, Ctime),
    sparams = c(.2, .4, .3, .3))
mmp(fit, smoother = "supsmu", incvar = F)
mmp(fit, smoother = "supsmu", h = cbind(AN, Btemp, Ctime),
    sparams = c(.6, .6, .6, .6), incvar = F)
```

Finally, clean up:

```
detach("naph")
rm(fit, naph, x) # S-PLUS
rm(fit, naph) # R
```

The following examples again use the "Wisconsin Breast Cancer" data:

```
wbcd <- read.table("data/WBCD.dat", header = T,
                   row.names = NULL)
attach(wbcd)
fit <- glm(Class1 ~ Adhes + BNucl + Chrom + NNucl + Thick,
           family = binomial)
```

All the examples above should work for this model also, for example:

```
mmp(fit, h = Mitos, sptry = c(4, 8, 12, 16), incvar = F)
mmp(fit, default = F, random = 2, incvar = F)
mmp(fit, h = cbind(Mitos, Adhes, BNucl),
    sparams = c(12, 4, 4, 4), incvar = F)
mmp(fit, h = Mitos, sparams = c(12, 4), incvar = F,
    included = names(Mitos[Mitos<10])) # S-PLUS
mmp(fit, h = Mitos, sparams = c(12, 4), incvar = F,
    included = as.character(seq(along=Mitos))[Mitos<10]) # R
mmp(fit, smoother = "loess1", sptry = c(.4, .3, .2, .1),
    incvar = F)
mmp(fit, smoother = "loess1", incvar = F,
    h = cbind(Mitos, Adhes, BNucl),
    sparams = c(.1, .95, .7, .7))
mmp(fit, smoother = "loess2", sptry = c(.4, .3, .2, .1),
    incvar = F)
mmp(fit, smoother = "loess2", incvar = F,
    h = cbind(Mitos, Adhes, BNucl),
    sparams = c(.3, .9, .85, .9))
mmp(fit, smoother = "kernel", sptry = c(.4, .3, .2, .1),
    incvar = F)
mmp(fit, smoother = "kernel", incvar = F,
    h = cbind(Mitos, Adhes, BNucl),
    sparams = c(.1, .3, .3, .3))
mmp(fit, smoother = "supsmu", sptry = c(.4, .3, .2, .1),
    incvar = F)
mmp(fit, smoother = "supsmu", incvar = F,
    h = cbind(Mitos, Adhes, BNucl),
    sparams = c(.1, .1, .1, .1), incvar = F)
```

Finally, clean up:

```
detach("wbcd")
rm(fit, wbcd)
```

5 Error messages and potential problems

The following error checks and warnings have been included:

- *Error: Smoother x not defined, try spline, loess1, loess2, kernel, or supsmu*
  The argument `smoother` must match one of `"spline"`, `"loess1"`, `"loess2"`, `"kernel"` or `"supsmu"`.

- *Warning: variance smooths not available for this smoother*
  Variance smooths cannot be calculated for `"supsmu"` because this smoother excludes duplicate x-values from its output.

- *Warning: BDM calculation not available for this smoother*
  The Bayes discrepancy measure cannot be calculated for `"supsmu"` because this smoother excludes duplicate x-values from its output.

- *Warning: variance smooths not needed for binary logistic*
  In models with a binary response, the variance is completely determined by the probability of success, regardless of the model, and so variance function checks are not applicable.

- *Error: You cannot specify sparams or spar and set spdefault = T*
  Only set `spdefault` equal to `T` if you want the default values for the smoothing parameters; otherwise specify them using `sparams` or `spar` or choose them interactively by omitting these arguments.

- *Error: spar must have the same number of components as h*
  The number of smoothing parameters specified must be equal to the number of plots required; the number of plots is given by the number of components of `h`, plus one if `default` equals `T`, plus the number of random directions specified (if any).

- *Error: sparams must have the same number of components as h*
  The number of smoothing parameters specified must be equal to the number of plots required; the number of plots is given by the number of components of `h`, plus one if `default` equals `T`, plus the number of random directions specified (if any).

- *Error: No action*
  This message may be returned if you exit the function "early", that is before all possible calculations have been completed—it does not mean that an error has occurred.

Other possible problems include the following:

- *Warning: lines out of bounds*
  Some of the smooths extend beyond the plot axes—try adjusting `mult` to accommodate all the smooths.

- *S-PLUS hangs*
  Occasionally S-PLUS on Linux may hang during the smoothing calculations—try adjusting the smoothing parameters or using an alternative smoothing method; this only seems to be a problem with S-PLUS 5.x, so upgrade to S-PLUS 6.x or try R if all else fails!

- *Error: couldn't find function "smooth.spline"*
  R will return an error like this if you forget to attach the `modreg` library before using `bmmp` or `mmp`.

- *Error: problem in trellis.par.set*
  S-PLUS will return an error like this if you try to construct a Trellis plot on a traditional graphics device; remember to set `usetrellis` to `F` if using traditional graphics on S-PLUS.

6 Using Hastie and Tibshirani's `gibbs.gam` function to sample from GAMs

This section provides a brief outline of how to use Hastie and Tibshirani's `gibbs.gam` function to obtain the posterior samples for the additive model example introduced in Section 4.1. See Hastie and Tibshirani (2000) for details on obtaining and using the `gibbs.gam` function.

This example again uses the "naphthalene" data. Read in the data and fit an *additive model* using the following (S-PLUS for UNIX/Linux only, since Hastie and Tibshirani's `gibbs.gam` function is currently only available for this platform):

```
naph <- read.table("data/naphthalene.dat", header = T,
                   row.names = NULL)
attach(naph)
x <- .397*AN + .445*Ctime + .802*Btemp
fit <- gam(Yn ~ s(x, df = 4))
```

Next, generate posterior samples using the `gibbs.gam` function (here `nwarm` is the number of warm-up samples to be discarded, `nkeep` is the number of subsequent samples to retain, and `var.comp` indicates whether variance component sampling is done or not—see Hastie and Tibshirani 2000 for further details):

```
nsamp <- 100
ggnaph <- gibbs.gam(fit, nwarm = 200, nkeep = nsamp,
                    var.comp = F)
```

Output in the command window indicates the samples being generated, first the warm-up samples, then the ones to be retained. Once the sampling is completed, `ggnaph` can be used as an input to `postsamp.gam` as described in Section 4.1.

To save this object for use later use `dump` and to read it in at a later date use `source`:

```
dump("ggnaph", fileout = "data/ggnaph")
source("data/ggnaph")
```

7   Using **BUGS** and **BOA** to sample from GLMs

This section provides a brief outline of how to use **BUGS** and **BOA** to obtain the posterior samples for the "Wisconsin Breast Cancer" example introduced in Section 4.1. See Spiegelhalter et al. (1999) for details on obtaining and using **BUGS** and Smith (2001) for details on obtaining and using **BOA**.

7.1   **BUGS**

The following is based on "WinBUGS 13". A script (`umwbcdodc`) containing the code given below is available at:

http://lcb1.uregon.edu/ipardoe/research/bmmpsoft.htm

- After starting **BUGS**, open a new `.odc` compound document file containing the following model specification (this is the same model as specified in Section 4.4):

```
model
  {
  for( i in 1:N ) {
    Class1[i] ~ dbern(p[i])
    logit(p[i]) <- beta.0.star +
      beta.1*(Adhes[i] - mean(Adhes[])) +
      beta.2*(BNucl[i] - mean(BNucl[])) +
      beta.3*(Chrom[i] - mean(Chrom[])) +
      beta.4*(NNucl[i] - mean(NNucl[])) +
      beta.5*(Thick[i] - mean(Thick[]))
    }
  beta.0 <- beta.0.star - beta.1*mean(Adhes[]) -
    beta.2*mean(BNucl[]) - beta.3*mean(Chrom[]) -
    beta.4*mean(NNucl[]) - beta.5*mean(Thick[])
  beta.0.star ~ dnorm(0.0,0.000001)
  beta.1 ~ dnorm(0.0,0.000001)
  beta.2 ~ dnorm(0.0,0.000001)
  beta.3 ~ dnorm(0.0,0.000001)
```

```
    beta.4 ~ dnorm(0.0,0.000001)
    beta.5 ~ dnorm(0.0,0.000001)
    }
```

- Open the `Model>Specification` dialog, double-click the word "model" in the compound document file, and click "check model" ("model is syntactically correct" should appear at the bottom left of the BUGS window).

- Add the data to the compound document file (to save space, only three of the 681 observations are shown below):

```
list( N=681)

Class1[] Adhes[] BNucl[] Chrom[] NNucl[] Thick[]
1 1 1 3 1 5
1 5 10 3 2 5
...
0 5 5 10 4 4
```

- Next, double-click the word "list" above, click "load data" ("data loaded" should appear), double-click "Class1" above, and click "load data" again ("data loaded" should appear again). Change "num of chains" to 2, and click "compile" (after a brief pause, "model compiled" should appear).

- Add initial values to the compound document file:

```
list(beta.0.star=0, beta.1=0, beta.2=0, beta.3=0,
     beta.4=0, beta.5=0)
list(beta.0.star=1.3, beta.1=-.4, beta.2=-.5,
     beta.3=-.6, beta.4=-.4, beta.5=-.8)
```

- Next, double-click the first "list" above, and double-click "load inits" (for chain 1) ("initial values loaded; model contains uninitialized nodes" should appear). Finally, double-click the second "list" below, and double-click "load inits" (for chain 2) ("initial values loaded; model initialized" should appear).

- Open the `Inference>Samples` dialog, and type "beta.0", click "set", type "beta.1", click "set", ..., type "beta.5", click "set". Then type "*" and click "trace" (a window titled "Dynamic trace" should open with six currently empty plots).

- Open the `Model>Update` dialog, and click "update" (the dynamic traces should begin to appear, until after a few minutes 1000 samples for each chain should have been generated).

30

- Click "history", "density", "stats", "quantiles", "GR diag", "autoC" to see various summaries of the chains. Click "coda" to create files to export into CODA or BOA (in S-PLUS or R)—CODA is another MCMC diagnostic tool, similar to BOA. Save "CODA index" (twice) as text-files "WBCD1.ind" and "WBCD2.ind", and "CODA for chain 1" as text-file "WBCD1.out" and "CODA for chain 2" as text-file "WBCD2.out".

7.2 BOA

The following is based on "BOA 0.5.0".

- After starting BOA (in S-PLUS or R), select "1:File", then "1:Import Data", then "1:BUGS Output File", then "WBCD1" (say) to read in the data from chain 1 saved above. Repeat for the data from chain 2.

- Next, select "7:Return to Main Menu", then "2:Data", then "7:Subset" to discard an initial part of each chain. For example, retain all chain indices and all parameters (just hit "Enter" twice), but for iterations type "501:1000" to discard the first half of each chain. To check the status of the current "Working Dataset" select "4:Display Working Dataset".

- Next, select "8:Return to Main Menu", then "3:Analysis" to obtain descriptive statistics and convergence diagnostics for the samples. And select "4:Return to Main Menu", then "4:Plot" to obtain descriptive and diagnostic plots.

- Finally, once you are satisfied with the samples in the "Working Dataset", select "4:Return to Main Menu", then "1:File". then "3:Save Session", then type the name of the object to which to save the session data, for example "boawbcd". Select "5:Exit BOA" to return to the S-PLUS or R prompt. Now boawbcd should be an object in your working directory, ready to be used as an input to postsamp.binlog as described in Section 4.1.

- To save this object for use later use data.dump (or dump if that does not work) (S-PLUS) or save (R), and to read it in at a later date use data.restore (or source if you used dump) (S-PLUS) or load (R):

```
data.dump("boawbcd", file = "data/boawbcd") # S-PLUS
data.restore("data/boawbcd")
dump("boawbcd", fileout = "data/boawbcd") # S-PLUS alt.
source("data/boawbcd")
save("boawbcd", file="data/boawbcd") # R
load("data/boawbcd")
```

References

Becker, R. A. and W. S. Cleveland (1996). *S-PLUS Trellis Graphics User's Manual*. Murray Hill, NJ: Bell Labs.

Casella, G. and E. I. George (1992). Explaining the Gibbs sampler. *The American Statistician 46*, 167–174.

Cleveland, W. S. and S. J. Devlin (1988). Locally-weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association 83*, 596–610.

Cook, R. D. and I. Pardoe (2000). Comment on "Bayesian backfitting" by T. J. Hastie and R. J. Tibshirani. *Statistical Science 15*, 213–216.

Friedman, J. H. (1984). A variable span scatterplot smoother. Technical Report 5, Laboratory for Computational Statistics, Stanford University.

Hastie, T. J. and R. J. Tibshirani (1990). *Generalized Additive Models*. Boca Raton, FL: Chapman & Hall/CRC.

Hastie, T. J. and R. J. Tibshirani (2000). Bayesian backfitting (with discussion). *Statistical Science 15*, 196–223.

McCullagh, P. and J. A. Nelder (1989). *Generalized Linear Models* (2nd ed.). Boca Raton, FL: Chapman & Hall/CRC.

Pardoe, I. (2001). A Bayesian sampling approach to regression model checking. *Journal of Computational and Graphical Statistics 10*, 617–627.

Pardoe, I. and R. D. Cook (2002). A graphical method for assessing the fit of a logistic regression model. *The American Statistician 56*. In press.

Ruppert, D., M. P. Wand, U. Holst, and O. Hössjer (1997). Local polynomial variance-function estimation. *Technometrics 39*, 262–273.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Boca Raton, FL: Chapman & Hall/CRC.

Smith, B. J. (2001). Bayesian Output Analysis Program (BOA). Version 1.0.0 for S-PLUS and R, available at http://www.public-health.uiowa.edu/boa.

Spiegelhalter, D. J., A. Thomas, and N. G. Best (1999). *WinBUGS Version 1.2 User Manual*. Cambridge, UK: MRC Biostatistics Unit.