Bayesian semiparametric regression based on mixed model methodology: A tutorial

Thomas Kneib, Stefan Lang and Andreas Brezger

Department of Statistics, University of Munich.

February 19, 2004

Abstract

This tutorial demonstrates the usage of *BayesX* for analysing Bayesian semiparametric regression models based on mixed model methodology. As an example we consider data on undernutrition of children in Zambia. The tutorial is designed to be self-contained and describes all features of *BayesX* in detail, that will be needed throughout the tutorial. Therefore it may also serve as a first introduction into the general usage of *BayesX*.

Contents

1

1 Introduction

This tutorial demonstrates the usage of *BayesX* for analysing Bayesian semiparametric regression models based on mixed model methodology. As an example we consider data on undernutrition of children in Zambia. This data has already been analysed in Kandala et al. (2001) and we will use the same model that has been developed there. Since our focus is on demonstrating how regression models can be estimated in *BayesX*, we do not discuss or interpret the estimation results but simply give the commands to produce them.

The main focus in this tutorial is on empirical Bayes inference based on mixed model methodology. *BayesX* also supports the estimation of semiparametric regression models in a full Bayesian context based on MCMC techniques. In the current implementation the empirical Bayes approach may be much faster in situations with a relatively small number of regression parameters and non-normal responses but the full Bayesian approach can deal with massive data sets while the empirical Bayes approach is limited to data sets with medium sample size. A further advantage of the empirical Bayes approach is, that questions about the convergence of MCMC samples or sensitivity on hyperparameters do not arise. A comparison of both approaches in a simulation study has shown, that the empirical Bayes approach yields somewhat better point estimates, especially for Bernoulli distributed response, see Fahrmeir, Kneib and Lang (2003).

A second tutorial, dealing with the full Bayesian approach is available from the tutorials section of the *BayesX*-homepage. All tutorials are designed to be self-contained and describe all features of *BayesX* in detail, that will be needed throughout the tutorial. Users who are already familiar with the usage of *dataset* and *map objects* may therefore skim through sections 3-5.

The theoretical background of Bayesian semiparametric regression will not be described in this tutorial. Chapter 7 of the manual may serve as a first introduction, further details about the estimation techniques for the empirical Bayes approach can be found in Fahrmeir, Kneib and Lang (2003). The full Bayesian approach is described in full detail in Fahrmeir and Lang (2001a,2001b), Lang and Brezger (2003) and Brezger and Lang (2003). Survival models are treated in Hennerfeind, Brezger and Fahrmeir (2003) and Fahrmeir and Hennerfeind (2003), Count data regression is covered in Fahrmeir and Osuna (2003).

2 Description of the data set

Undernutrition among children is usually determined by assessing the anthropometric status of a child relative to a reference standard. In our example undernutrition is measured by stunting or insufficient height for age, indicating chronic undernutrition. Stunting for a child i is determined using a Z-score which is defined as

$$Z_i = \frac{AI_i - MAI}{\sigma}$$

where AI refers to the child's anthropometric indicator (height at a certain age in our example), MAI refers to the median of the reference population and σ refers to the standard deviation of the reference population.

The main interest is on modelling the dependence of undernutrition on covariates including the age of the child, the body mass index of the child's mother, the district the child lives in and some further categorial covariates. Table 1 gives a description of the variables that we will use in our model.

Variable	Description
$hazstd$	standardised Z-score of stunting
bmi	body mass index of the mother
agc	age of the child in months
$district$	district where the child lives
rcw	mother's employment status with categories "working" $(= 1)$ and "not working" $(= -1)$
$edu1/2$	mother's educational status with categories "complete primary but incomplete secondary" $(edu1 = 1)$, "complete secondary or higher" $(edu2 = 1)$ and "no education or incomplete primary" $(edu1 = edu2 = -1)$
tpr	locality of the domicile with categories "urban" $(= 1)$ and "rural" $(= -1)$
sex	gender of the child with categories "male" $(= 1)$ and "female" $(= -1)$

Table 1: Variables in the undernutrition data set.

3 Getting started

After having started *BayesX*, a main window with four sub-windows appears on the screen. These are a *command window* for entering and executing code, an *output window* for displaying results, a *review window* for easy access to past commands, and an *object browser* that displays all objects currently available.

BayesX is object oriented although the concept is limited, i.e. inheritance and other concepts of object oriented languages like C++ or S-plus are not supported. For every object type a number of object-specific methods may be applied to a particular object. The syntax for generating a new object in *BayesX* is

> *objecttype objectname*

where *objecttype* is the type of the object, e.g. `dataset`, and *objectname* is the name to be given to the new object.

The rest of the tutorial is separated in five parts dealing with the different steps of estimating a regression model. In section 4 we create a *dataset object* to incorporate, handle and manipulate the data. We will also give a brief description of some methods that may be applied to *dataset objects*. Since we want to estimate a spatial effect of the district in which a child lives, we need the boundaries of the districts to compute the neighbourhood information of the map of Zambia. This information will be stored in a *map object*. Section 5 describes how to create and handle these objects. Estimation of the regression model is carried out in section 6 using a *remlreg object*. The last two sections describe how to visualise the estimation results and how to customise the obtained graphics.

If you have not done so yet, please download the data set and the *boundary file* associated with this tutorial now. You may also want to download the batch file containing the commands used in the following sections. Please note, that paths within these commands must be changed according to the storage location of the corresponding files on your hard disk.

4 Reading data set information

In a first step we read the available data set information into *BayesX*. Therefore we create a *dataset object* named d:

> `dataset d`

We store the data in d using the method `infile`:

```
> d.infile, maxobs=5000 using c:\data\zambia.raw
```

Note, that we assume the data to be provided in the external file `c:\data\zambia.raw`. The first few lines of this file look like this:

```
hazstd bmi agc district rcw edu1 edu2 tpr sex
0.0791769   21.83   4    81   -1   1   0   1   -1
-0.2541965  21.83   26   81   -1   1   0   1   -1
-0.1599823  20.43   56   81   1    -1  -1  1   1
0.1733911   22.27   6    81   -1   0   1   1   1
```

In our example the file contains the variable names in the first line. Therefore it is not necessary to specify them in the `infile` command. If the file contained only the data without variable names, we would have to supply them after the keyword `infile`:

```
> d.infile hazstd bmi agc district rcw edu1 edu2 tpr sex, maxobs=5000
    using c:\data\zambia.raw
```
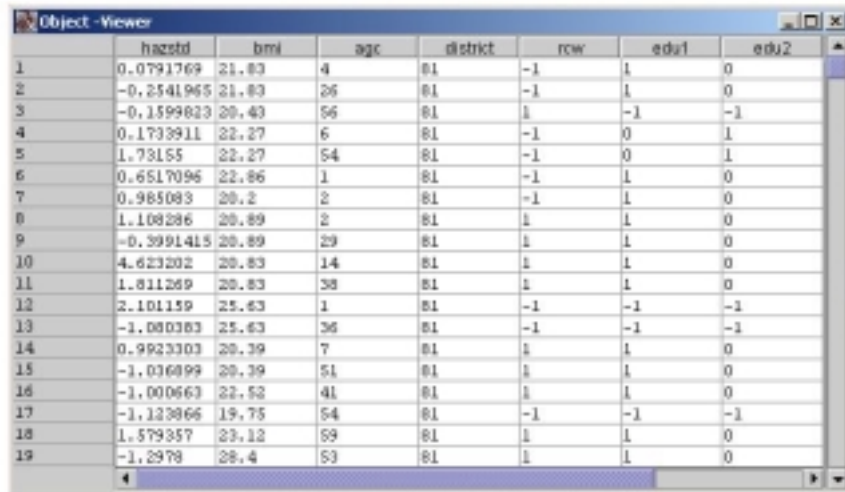
Option `maxobs` can be used to speed up the execution time of the `infile` command. If `maxobs` is specified, *BayesX* allocates enough memory to store all the data while the total amount of required memory is unknown in advance if `maxobs` remains unspecified. For larger data sets this may cause *BayesX* to start reading the data set information several times because the currently allocated memory is exceeded. However, this is only meaningful for larger data sets with more than 10,000 observations and could therefore be omitted in our example.

A second option that may added to the `infile` command is the `missing` option to indicate missing values. Specifying for example 'missing = M' defines the letter 'M' as an indicator for a missing value. The default for missing values are a period '.' and 'NA' (which remain valid indicators for missing values even if an additional indicator is defined by the `missing` option).

After having read in the dataset information we can inspect the data visually. Executing the command

```
> d.describe
```

opens an *Object-Viewer* window containing the data in form of a spreadsheet (see Figure 1). This can also be achieved by double-clicking on the *dataset object* in the *object browser*.

Figure 1: A screenshot of the dataset.

4

Further methods allow to examine the variables in the *dataset object*. For a categorial variable, e.g. *sex*, the `tabulate` command may be used to produce a frequency table:

```
> d.tabulate sex
```

resulting in

```
Variable: sex

        Value        Obs        Freq         Cum
           -1       2451      0.5057      0.5057
            1       2396      0.4943           1
```

being printed in the *output window*. For continuous variables the `descriptive` command prints several characteristics of the variable in the output window. E.g., executing

```
> d.descriptive bmi
```

leads to

```
Variable    Obs       Mean    Median        Std       Min       Max
     bmi   4847  21.944349      21.4  3.2879659      12.8     39.29
```

5   Map objects

In the following we want to estimate a spatially correlated effect of the district in which a child lives. Therefore we need the boundaries of the districts in Zambia to compute the neighbourhood information of the map of Zambia. We therefore create a *map object*

```
> map m
```

and read in the boundaries using the `infile` command of *map objects*:

```
> m.infile using c:\data\zambia.bnd
```

Having read in the boundary information, *BayesX* automatically computes the neighbourhood matrix of the map.

The file following the keyword `using` is assumed to contain the boundaries in form of closed polygons. To give an example we print a small part of the boundary file of Zambia. The map corresponding to the section of the boundary file can be found in Figure 2.

> ⋮

```
"52",48
28.080507,-12.537530
28.083376,-12.546980
28.109501,-12.548961
28.134972,-12.566787
28.154797,-12.585320
28.165771,-12.593912
28.165771,-12.593912
28.160769,-12.609917
28.152800,-12.633824
28.144831,-12.657733
28.132877,-12.677656
28.120922,-12.701565
28.120922,-12.717505
28.120922,-12.741411
28.116938,-12.761335
28.108969,-12.777274
28.100998,-12.793213
28.089045,-12.817122
```

```
28.085060,-12.837045
28.081076,-12.856968
28.081076,-12.876892
28.080862,-12.884153
28.080862,-12.884153
28.076630,-12.879521
28.031454,-12.881046
27.974281,-12.884675
27.910725,-12.878692
27.686228,-12.880120
27.665676,-12.854732
27.653563,-12.818301
27.639263,-12.759848
27.648254,-12.699927
27.662464,-12.680613
27.662464,-12.680613
27.666534,-12.675080
27.703260,-12.679779
27.752020,-12.695455
27.797932,-12.702188
27.836775,-12.707567
27.867813,-12.699892
27.902308,-12.667418
27.922668,-12.630853
27.943035,-12.596350
27.963434,-12.571486
27.983179,-12.563844
28.016331,-12.554779
28.070650,-12.542199
28.080507,-12.537530
```

$\vdots$

For each region of the map the boundary file must contain the identifying name of the region, the polygons that form the boundary of the region, and the number of lines the polygon consists of. The first line always contains the region code surrounded by quotation marks and the number of lines the polygon of the region consists of. The code and the number of lines must be separated by a comma. The subsequent lines contain the coordinates of the straight lines that form the boundary of the region. The straight lines are represented by the coordinates of their end points. Coordinates must be separated by a comma. Note that the first and the last point must be identical (see the example above) to obtain a closed polygon. Compare chapter 5 of the complete manual for a detailed description of some special cases, e.g. regions divided into subregions.
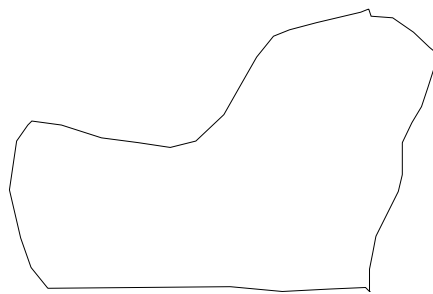

Figure 2: Corresponding graph of the section of the boundary file

Map objects may be visualised using method `describe`:

```
> m.describe
```

resulting in the graph shown in Figure 3. Additionally, `describe` prints further information about the *map object* in the *output window* including the name of the object, the number of regions, the

minimum and maximum number of neighbours and the bandwidth of the corresponding adjacency or neighbourhood matrix:

```
MAP m
Number of regions: 54
Minimum number of neighbors: 1
Maximum number of neighbors: 9
Bandsize of corresponding adjacency matrix: 24
```
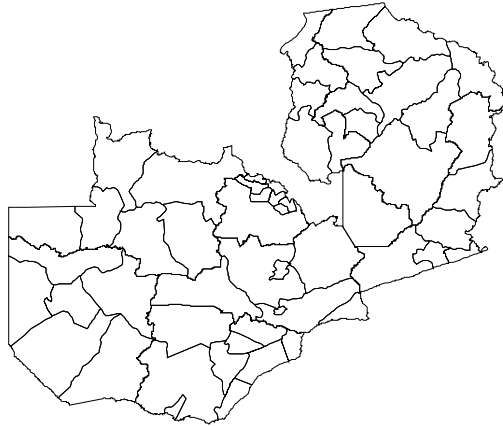

Figure 3: The districts within Zambia.

Reading the boundary information from an external file and computing the neighbourhood matrix may be a computationally intensive task if the map contains a huge number of regions or if the polygons are given in great detail. To avoid doing these computations every time *BayesX* is restarted, we may store the computed neighbourhood information in a so-called *graph file* to make it directly available. To do this, we use the method `outfile` together with the `graph` option as in the following example:

```
> m.outfile, replace graph using c:\data\zambia.gra
```

Note, that specifying the option `replace` allows *BayesX* to overwrite an existing file with the same name. Without this option an error message would be raised if the given file is already existing. Leaving out the keyword `graph` in the above command leads to storage of the map in boundary format.

A graph file stores the nodes and the edges of a graph $G = (N, E)$, see for example George and Liu (1981, Ch. 3) for a first introduction into graph theory. A graph is a convenient way of representing the neighbourhood structure of a geographical map. The nodes of the graph correspond to the region codes. The neighbourhood structure is represented by the edges of the graph. In some situations it may be useful to define weights associated with the edges of a graph which can be be stored in the *graph file* as well.

We now describe the structure of a graph file as it is expected by *BayesX*. The first line of a *graph file* must contain the total number of nodes of the graph. In the remaining lines, the nodes of the graph together with their edges and associated weights are specified. One node corresponds to three consecutive lines. The first of the three lines must contain the name of the node, which may simply be the name of a geographical region. In the second line the number of edges of that particular node is given. The third line contains the corresponding edges of the node, where an edge is given by the index of a neighbouring node. The index starts with zero. For example, if the fourth and the seventh node/region in the *graph file* are connected/neighbours, the edge index for the fourth node/region is 6 and for the seventh node/region 3.

We illustrate the structure of a graph file with an example. The following few lines are the beginning of the graph file corresponding to the map of Zambia:

```
57
11
4
1 2 4 5
12
4
0 3 5 6
13
2
0 4
```

⋮

The first line specifies the total number of nodes, in the present example 57 nodes. The subsequent three lines correspond to the node with name '11', which is the first region in the map of Zambia. Region '11' has 4 neighbours, namely the second, third, fifth and sixth node appearing in the graph file. Once again, note that the index starts with zero, i.e. 0 corresponds to the first node, 1 corresponds to the second node and so on. Lines 5 to 7 in the example correspond to node '12' and its four neighbours and lines 8 to 10 correspond to node '13'.

In a graph file it is also possible to specify weights associated with the edges of the nodes. Since in the preceding example no weights are explicitly specified, all weights are automatically defined to be equal to one. Nonequal weights are specified in the graph file by simply adding them following the edges of a particular node. An example of the beginning of a graph file with weights is given below:

```
57
11
4
1 2 4 5 0.461261 1.74605 1.13411 1.17406
12
4
0 3 5 6 0.461261 0.388206 0.537407 0.756847
13
2
0 4 1.74605 1.1692
```

⋮

Here the edges of the first node '11' have weights 0.461261, 1.74605, 1.13411 and 1.17406.

Note, that graph files allow the estimation of more general Markov random fields. While the polygons stored in a *boundary file* represent geographical information, the nodes and edges of a graph may define arbitrary neighbourhood structures. For example, the definition of 3 dimensional Markov random fields representing space-time interactions is possible.

To see how storing maps in *graph files* affects the computation time of the `infile` command, we create a second *map object* and read in the information from the graph file. Again, we have to specify the keyword `graph`:

```
> map m1
> m1.infile, graph using c:\data\zambia.gra
```

As you should have noticed, reading geographical information from a *graph file* is usually much faster than reading from a *boundary file*. However, using *graph files* also has a drawback. Since they

do no longer contain the full information on the polygons forming the map, we can not visualise a *map object* created from a *graph file*. Trying to do so

```
> m1.describe
```

raises an error message. This implies, that visualising estimation results of spatial effects can only be based on *map objects* created from *boundary files*, although estimation can be carried out using *graph files*. Since we will work with the *map object* m in the following, we delete m1:

```
> drop m1
```

6   Bayesian semiparametric regression

To estimate a regression model based on mixed model techniques, we first create a *remlreg object*:

```
> remlreg r
```

By default estimation results are written to the subdirectory `output` of the installation directory. In this case the default filenames are composed of the name of the *remlreg object* and the type of the specific file. Usually it is more convenient to store the results in a user-specified directory. To define this directory we use the `outfile` command of *remlreg objects*:

```
> r.outfile = c:\data\r
```

Note, that `outfile` does not only specify a directory but also a base filename (the character 'r' in our example). Therefore executing the command above leads to storage of the results in the directory 'c:\data' and all filenames start with the character 'r'. Of course the base filename may be different from the name of the *remlreg object*.

In addition to parameter estimates *BayesX* also produces some further information on the estimation process. In contrast to parameter estimates this information is not stored automatically but is printed in the *output window*. Therefore it is useful to store the contents of the *output window*. This can be achieved automatically by opening a *log file* using the `logopen` command

```
> logopen, replace using c:\data\logreml.txt
```

After opening a *log file*, every information written to the output window is also stored in the log file. Option `replace` allows *BayesX* to overwrite an existing file with the same name as the specified *log file*. Without `replace` results are appended to an existing file.

The model presented in Kandala et al. (2001) is given by the following semiparametric predictor:

$$\eta = \gamma_0 + \gamma_1 rcw + \gamma_2 edu1 + \gamma_3 edu2 + \gamma_4 tpr + \gamma_5 sex + f_1(bmi) + f_2(agc) + f^{str}(district) + f^{unstr}(district)$$

The two continuous covariates *bmi* and *agc* are assumed to have a possibly nonlinear effect on the Z-score and are therefore modelled nonparametrically (as P-splines with second order random walk prior in our example). The spatial effect of the district is split up into a spatially correlated part $f^{str}(district)$ and an uncorrelated part $f^{unstr}(district)$, see Fahrmeir and Lang (2001b) for a motivation. The correlated part is modelled by a Markov random field prior, where the neighbourhood matrix and possible weights associated with the neighbours are obtained from the *map object* m. The uncorrelated part is modelled by an i.i.d Gaussian effect.

To estimate the model we use method `regress` of *remlreg objects*:

```
> r.regress hazstd = rcw + edu1 + edu2 + tpr + sex + bmi(psplinerw2)
  + agc(psplinerw2) + district(spatial,map=m) + district(random),
  family=gaussian lowerlim=0.01 eps=0.0005 using d
```

Options `lowerlim` and `eps` control the estimation process. Since small variances are near to the boundary of their parameter space, the usual Fisher-scoring algorithm for their determination has

to be modified. If the fraction of the penalised part of an effect relative to the total effect is less than `lowerlim`, the estimation of the corresponding variance is stopped and the estimator is defined to be the current value of the variance (see chapter 7 in the manual for details). The option `eps` defines the termination criterion for the estimation process. The default value for `lowerlim` is 0.001, the default value for `eps` is 0.00001. However, since our analysis is only for explanatory purpose we chose somewhat weaker conditions resulting in a faster 'convergence' of the algorithm. On a 2.4 GHz PC estimation of our model took about 2 minutes and 30 seconds with the above specifications of `lowerlim` and `eps`.

A further option of method `regress` is `maxit`, defining the maximum number of iterations that should be performed in the estimation. Note, that *BayesX* produces results based on the current values of all parameters even if no convergence could be achieved within `maxit` iterations, but a warning message will be printed in the *output window*.

In the following we reproduce the content of the *output window* to make the user familiar with the estimation results produced by *BayesX*:

```
ESTIMATION RESULTS:

  Estimated scale parameter: 0.802145

  Scale parameter is also stored in file
  c:\data\r_scale.res


  f_bmi_pspline

  Estimated variance:  1.14816e-05
  Inverse variance:    87095.9
  Smoothing parameter: 69863.5
  (Smoothing parameter = scale / variance)
  NOTE: Estimation of the variance was stopped after iteration 6
        because the corresponding penalized part was small relative to the linear predictor.

  Variance and smoothing parameter are stored in file
  c:\data\r_f_bmi_pspline_var.res

  Results are stored in file
  c:\data\r_f_bmi_pspline.res

  Postscript file is stored in file
  c:\data\r_f_bmi_pspline.ps

  Results may be visualized using method 'plotnonp'
  Type for example: objectname.plotnonp 1


  f_agc_pspline

  Estimated variance:  0.00322146
  Inverse variance:    310.418
  Smoothing parameter: 249
  (Smoothing parameter = scale / variance)

  Variance and smoothing parameter are stored in file
  c:\data\r_f_agc_pspline_var.res

  Results are stored in file
  c:\data\r_f_agc_pspline.res

  Postscript file is stored in file
  c:\data\r_f_agc_pspline.ps

  Results may be visualized using method 'plotnonp'
  Type for example: objectname.plotnonp 2
```

```
f_district_spatial

Estimated variance:  0.0294012
Inverse variance:    34.0123
Smoothing parameter: 27.2828
(Smoothing parameter = scale / variance)

Variance and smoothing parameter are stored in file
c:\data\r_f_district_spatial_var.res

Results are stored in file
c:\data\r_f_district_spatial.res

Postscript file is stored in file
c:\data\r_f_district_spatial.ps

Results may be visualized in BayesX using method 'drawmap'
Type for example: objectname.drawmap 3


f_district_random

Estimated variance:  0.00806668
Inverse variance:    123.967
Smoothing parameter: 99.4393
(Smoothing parameter = scale / variance)

Variance and smoothing parameter are stored in file
c:\data\r_f_district_random_var.res

Results for random effects are stored in file
c:\data\r_f_district_random.res


FixedEffects

Variable  Post. Mode    Std. Dev.     p-value        95% Confidence Interval
const     0.0610357     0.0341574     0.0367456      -0.00592622    0.127998
rcw       0.00767158    0.0136564     0.286931       -0.0191003     0.0344434
edu1      -0.0605105    0.0261369     0.0103181      -0.111749      -0.00927192
edu2      0.234917      0.0459925     4.41249e-06    0.144754       0.325081
tpr       0.0904093     0.0218891     6.17648e-05    0.047498       0.133321
sex       -0.0585716    0.0129304     2.04243e-05    -0.0839203     -0.0332229

Results for fixed effects are also stored in file
c:\data\r_FixedEffects.res


Additive predictor and expectations

Additive predictor and expectation for each observation are stored in file
c:\data\r_predict.raw

Files of model summary:

-------------------------------------------------------------------------

Batch file for visualizing effects of nonlinear functions is stored in file
c:\data\r_graphics.prg

NOTE: 'input filename' must be substituted by the filename of the boundary-file

-------------------------------------------------------------------------

Batch file for visualizing effects of nonlinear functions
in S-Plus is stored in file
c:\data\r_splus.txt
```

```
NOTE: 'input filename' must be substituted by the filename of the boundary-file

----------------------------------------------------------------------------

Latex file of model summaries is stored in file
c:\data\r_model_summary.tex

----------------------------------------------------------------------------
```

In addition to the information being printed to the *output window* results for each effect are written to external ASCII files. The names of these files are given in the output window, compare the previous pages. For the variance parameters the files contain the variance as well as the corresponding smoothing parameter. For the different terms of the model the files contain the posterior mode, the 80% and 95% credible interval, the standard deviations and the corresponding 95% and 80% posterior probabilities of the estimated effects. Note, that credible intervals and posterior probabilities are based on a normal approximation of the posterior. For example, the beginning of the file `c:\data\r_f_bmi_pspline.res` for the effect of *bmi* looks like this:

```
intnr  bmi    pmode     ci95lower  ci80lower  std        ci80upper  ci95upper  pcat95  pcat80
1      12.8   -0.22305  -0.304661  -0.276409  0.0416301  -0.169692  -0.141439  -1      -1
2      13.15  -0.215246 -0.292828  -0.26597   0.0395749  -0.164522  -0.137663  -1      -1
3      14.01  -0.19607  -0.264173  -0.240597  0.0347394  -0.151544  -0.127968  -1      -1
```

The credible intervals and posterior probabilities that are computed for every effect may be changed by the user using the options `level1` and `level2`. For example specifying `level1=99` and `level2=70` in the option list of the `regress` command leads to the computation of 70% and 99% credible intervals and posterior probabilities. The defaults are `level1=95` and `level2=80`.

Some nonparametric effects are visualised by *BayesX* automatically and the resulting graphs are stored in ps format. E.g. the effect of *bmi* is visualised in the file `c:\data\b_f_bmi_pspline.ps` (compare the results on the previous pages for the other filenames). A batch file to reproduce the plots is stored in the output directory. In our example the name of the file is `c:\data\r_graphics.prg`. The advantage is that additional options may be added by the user to customise the graphs (compare the following two sections).

Moreover a file with ending `.tex` is created in the outfile directory. This file contains a summary of the estimation results and may be compiled using LaTeX.

Having finished the estimation we may close the *log file* by typing

```
> logclose
```

Note, that the *log file* is closed automatically when you exit BayesX.

7   Visualising estimation results

BayesX provides three possibilities to visualise estimation results:

- As mentioned in the previous section, certain results are automatically visualised by *BayesX* and stored in *ps files*.

- Post estimation commands of *bayesreg objects* allow to visualise results after having executed a `regress` command.

- *Graph objects* may be used to produce graphics using the ASCII files containing the estimation results. In principle *graph objects* allow the visualisation of any content of a *dataset object*. *Graph files* are also used in the batch file containing the commands to reproduce the automatically generated graphics.

In this section we describe the general usage of the post estimation commands as well as the commands for the usage with *graph objects* to enable the user to reproduce the automatically generated plots directly in *BayesX*. Section 8 describes how to customise plots.

7.1 Post estimation commands

After having estimated a regression model plots for nonparametric effects of metrical covariates can be produced using the post estimation command `plotnonp`:

```
> r.plotnonp 1
```

and

```
> r.plotnonp 2
```

produce the graphs shown in Figure 4 in an *object-viewer window*. The numbers following the `plotnonp` command depend on the order in which the model terms have been specified. The numbers are supplied in the *output window* after estimation, compare the results in the previous section.

By default the plots contain the posterior mode and pointwise credible intervals according to the levels specified in the `regress` command. So by default the plots include pointwise 80% and 95% credible intervals.
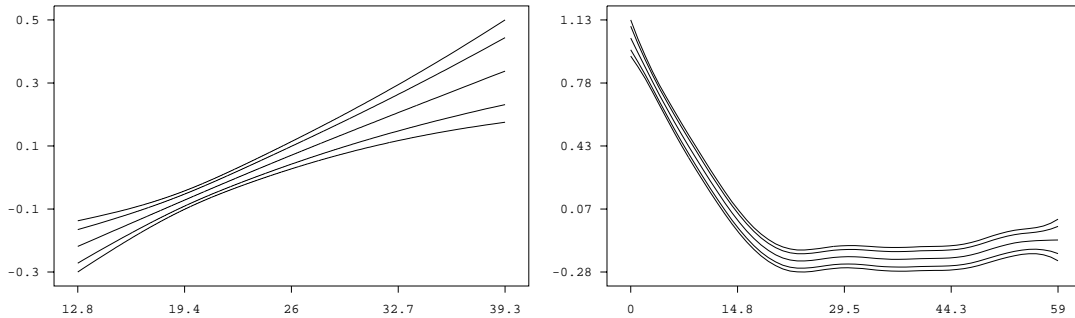

Figure 4: Effect of the body mass index of the child's mother and of the age of the child together with pointwise 80% and 95% credible intervals.

A plot may be stored in ps format using the `outfile` option. Executing

```
> r.plotnonp 1, replace outfile = c:\data\f_bmi.ps
```

stores the plot for the estimated effect of *bmi* in the file `c:\data\f_bmi.ps`. Again, specifying `replace` allows *BayesX* to overwrite an existing file. Note, that *BayesX* does not display the graph on the screen if the option `outfile` is specified.

Estimation results for spatial effects are best visualised by drawing the respective map and colouring the regions of the map according to some characteristic of the posterior, e.g. the posterior mode. For the structured spatial effect this can be achieved using the post estimation command `drawmap`

```
> r.drawmap 3
```

which results in the graph shown in Figure 5.

7.2 Graph Objects

The commands presented in the previous subsection work only after having estimated a regression model in the current BayesX session but it may also be useful to visualise results of former analyses.
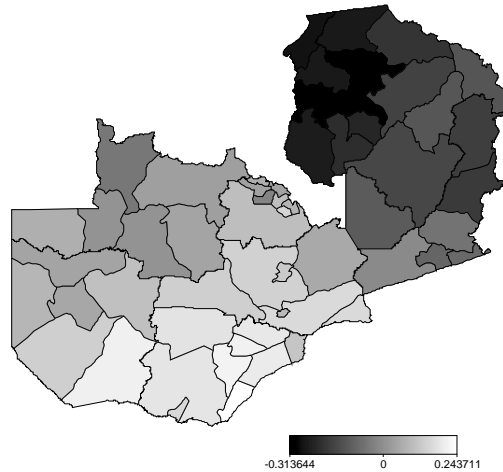
Figure 5: Posterior mode of the structured spatial effect.

This can be achieved using *graph objects*. Note again, that *graph files* are also used in the batch file containing the commands to reproduce the automatically generated graphics. Therefore the purpose of this subsection is also to enable the user to understand the content of this batch file.

First we read the estimation results into a *dataset object*. For example the estimation results for the effect of *bmi* can be read into *BayesX* by executing the commands

```
> dataset res
> res.infile using c:\data\r_f_bmi_pspline.res
```

Now the estimation results (or any content of a *dataset object*) may be visualised using a *graph object* which we create by typing

```
> graph g
```

The results stored in the *dataset object* `res` are now visualised using the `plot` command of *graph objects*. Executing

```
> g.plot bmi pmode ci95lower ci80lower ci80upper ci95upper using res
```

reproduces the graph in Figure 4.

Similar as for `plotnonp`, the direct usage of the `drawmap` command is only possible after executing a `regress` command. However, using *graph objects* again allows us to visualise results that have been stored in a file.

First we read the information contained in this file into a *dataset object*. For example the following command

```
> res.infile using c:\data\r_f_district_spatial.res
```

stores the estimation results for the structured spatial effect in the *dataset object* `res`. Now we can visualise the posterior mode using method `drawmap` of *graph objects* leading again to the graph shown in Figure 5:

```
> g.drawmap pmode district, map=m using res
```

Since – in contrast to a *remlreg object* – no *map object* is associated with a *graph object* we explicitly have to specify the map that we want to use in the option list.

Using *graph objects* also allows us to plot other characteristics of the posterior than the posterior mode. For instance the posterior 95% probabilities may be visualised by

```
> g.drawmap pcat95 district, map=m using res
```
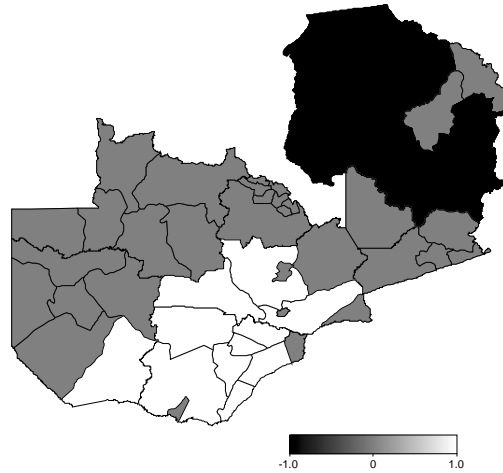
The result is shown in Figure 6.

Figure 6: Posterior 95% probability of the structured spatial effect.

A further advantage of *graph objects* is, that they allow to visualise the estimation results for the uncorrelated spatial effects. Since these are modelled as unstructured random effects, *BayesX* is unable to recognise them as spatial effects. However, proceeding as follows gives us the possibility to plot the unstructured spatial effect shown in Figure 7:

```
> res.infile using c:\data\r_f_district_random.res
> g.drawmap pmode district, map=m color swapcolors using res
```
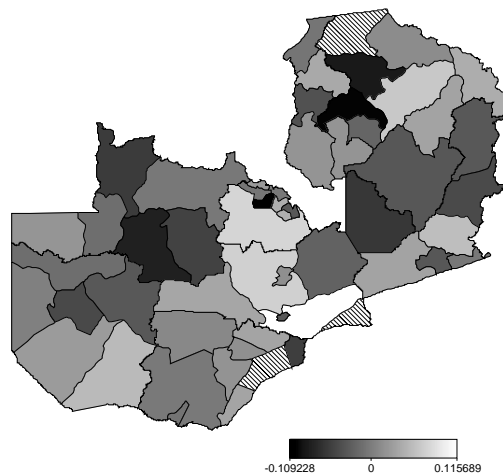

Figure 7: Posterior mode of the unstructured spatial effect.

8 Customising graphics

This section describes how to customise graphics created in *BayesX*. All options are described for the usage with the post estimation commands but may be used with graph files as well. So the options presented in this section also enable the user to modify the batch file containing the commands to reproduce the automatically generated graphics.

For the presentation of nonparametric effects it may be desirable to include only one of the credible interval into the plot. This is achieved by specifying the `levels` option. Possible values of this option are 1 and 2, corresponding to the levels specified in the `regress` command (compare section 6). If the default values of `level1` and `level2` have been used, specifying `level=2` in the `plotnonp` command causes *BayesX* to plot the 80% credible interval only (Figure 8):

```
> r.plotnonp 1, levels=2
```
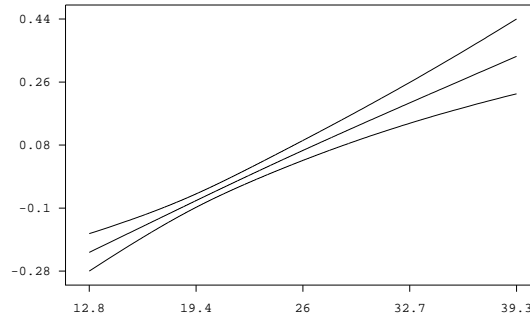

Figure 8: Effect of the body mass index of the child's mother with pointwise 80% credible intervals only.

It may be useful to add some more information to the graphs of nonparametric effects to distinguish more obviously between different covariates. Ways to do so are the specification of a title or the specification of axis labels. Both possibilities are supported by *BayesX* as demonstrated in the following examples (compare Figure 9 for the resulting plots):

```
> r.plotnonp 1, title="Mother body mass index"
> r.plotnonp 1, xlab="bmi" ylab="f_bmi" title="Mother body mass index"
```
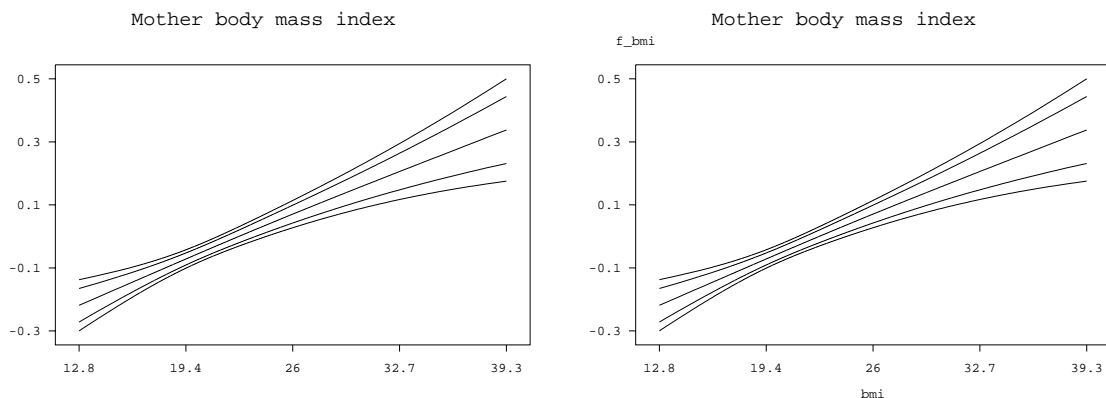

Figure 9: Specification of title and axis labels.

By default *BayesX* displays x- and y-axis with five equidistant ticks according to the range of the data that is to be visualised. These defaults may be overwritten using the options `xlimbottom`, `xlimtop` and `xstep` for the x-axis and `ylimbottom`, `ylimtop` and `ystep` for the y-axis, respectively. The usage of these options is more or less self-explanatory and is demonstrated in the following commands which lead to the graph shown in Figure 10.

```
> r.plotnonp 1, xlab="bmi" ylab="f_bmi" title="Mother body mass index"
  ylimbottom=-0.8 ylimtop=0.6 ystep=0.2 xlimbottom=12 xlimtop=40
```

Figure 10 also includes a graph for the effect of the age of the child that is customised in the same way as for the effect of *bmi*.

16

```
> r.plotnonp 2, xlab="age" ylab="f_age" title="Age of the child in months"
  ylimbottom=-0.3  ystep=0.3 xlimbottom=0 xlimtop=60 xstep=10
```
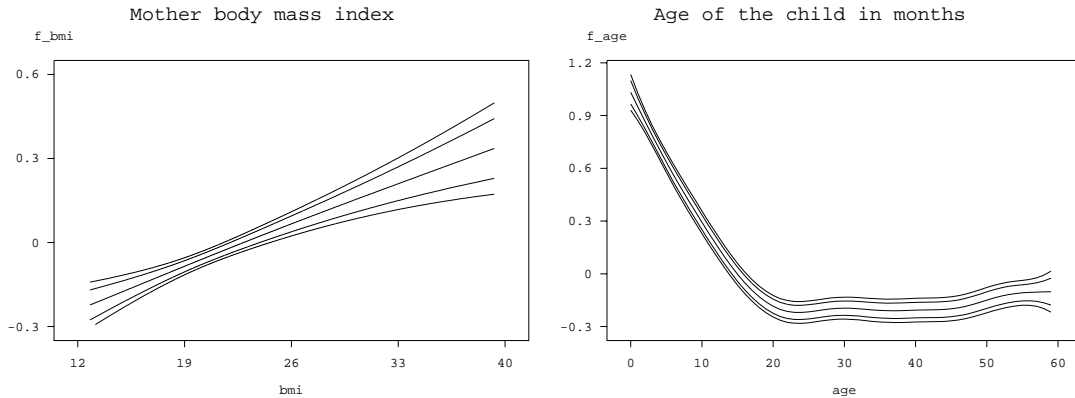

Figure 10: Re-defining x- and y-axis.

Now we turn to the options for method `drawmap`. By default `drawmap` uses grey scales to represent different values of the posterior mode. Using the option `color` forces *BayesX* to use different colours instead. Here the default would be to represent higher values through green colours and smaller values through red colours. Specifying `swapcolors` switches this definition. Therefore the following command

```
> r.drawmap 3, color swapcolors
```

leads to the graph shown in Figure 11 with higher values being represented through red colours and smaller values through green colours.
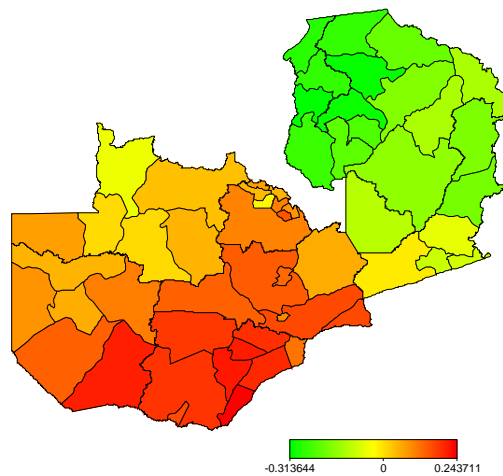

Figure 11: Posterior mode of the structured spatial effect in colour.

Similar options as for the visualisation of nonparametric effects exist for method `drawmap`. For example, a title may be included by specifying the option `title`

```
> r.drawmap 3, color swapcolors title="Structured spatial effect"
```

or the range of values to be displayed may be defined using the options `lowerlimit` and `upperlimit`:

```
> r.drawmap 3, color swapcolors title="Structured spatial effect" lowerlimit=-0.3
  upperlimit=0.3
```

17

The graph produced by the second command is shown in Figure 12.
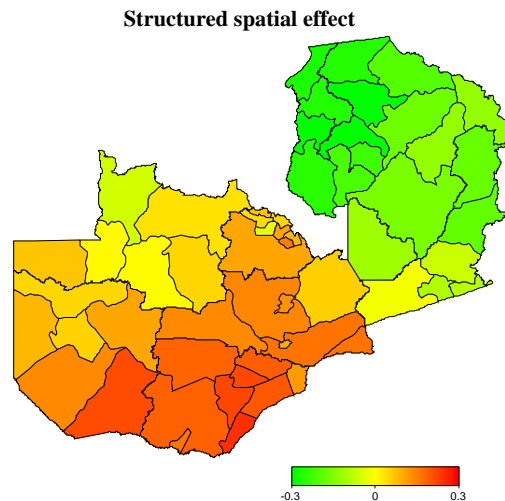
Structured spatial effect

Figure 12: Specifying a title and the range of the plot for spatial effects.

References

Brezger, A. and Lang, S., 2003: Generalized structured additive regression based on Bayesian P-splines. SFB 386 Discussion paper 321, Department of Statistics, University of Munich.

Fahrmeir, L. and Hennerfeind, A., 2003: Nonparametric Bayesian hazard rate models based on penalized splines. SFB 386 Discussion paper 361, University of Munich.

Fahrmeir, L., Kneib, T. and Lang, S., 2003: Penalized structured additive regression for space-time data: A Bayesian perspective, *Statistica Sinica (under revision)*. Available from www.stat.uni-muenchen.de/∼kneib/papers.html.

Fahrmeir, L. and Lang, S., 2001a: Bayesian Inference for Generalized Additive Mixed Models Based on Markov Random Field Priors. *Journal of the Royal Statistical Society C*, 50, 201-220.

Fahrmeir, L. and Lang, S., 2001: Bayesian Semiparametric Regression Analysis of Multicategorical Time-Space Data. *Annals of the Institute of Statistical Mathematics*, 53, 10-30

Fahrmeir, L. and Osuna, L. (2003), Structured count data regression. SFB 386 Discussion paper 334, University of Munich.

George, A. and Liu, J.W. 1981: *Computer Solution of Large Sparse Positive Definite Systems*, Prentice–Hall.

Hennerfeind, A., Brezger, A. and Fahrmeir, L., 2003: Geoadditive survival models. SFB Discussion paper 333, University of Munich.

Kandala, N. B., Lang, S., Klasen, S. and Fahrmeir, L. (2001): Semiparametric Analysis of the Socio-Demographic and Spatial Determinants of Undernutrition in Two African Countries. *Research in Official Statistics*, 1, 81-100.

Lang, S. and Brezger, A., 2003: Bayesian P-splines. *Journal of Computational and Graphical Statistics*, to appear.