BayesX

Software for Bayesian Inference in Structured Additive Regression Models

Version 2.00

Tutorials

Developed by

Christiane Belitz
Andreas Brezger
Thomas Kneib (Carl von Ossietzky University Oldenburg)
Stefan Lang (Leopold-Franzens-University Innsbruck)

With contributions by

Eva-Maria Fronk
Felix Heinzl
Andrea Hennerfeind
Manuela Hummel
Alexander Jerak
Susanne Konrath
Petra Kragler
Cornelia Oberhauser
Leyre Estíbaliz Osuna Echavarría
Daniel Sabanés Bové

Supported by

Ludwig Fahrmeir (mentally)
Leo Held (mentally)
German Science Foundation

Acknowledgements

using a Z-score 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), while MAI and σ correspond to the median and the standard deviation in the reference population, respectively.

Our 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 1.1 gives a description of the variables that we will use in our model.

Variable	Description
hazstd	standardised Z-score for stunting
bmi	body mass index of the mother
agc	age of the child in months
district	district where the mother lives
rcw	mother's employment status with categories "working" (= 1) and "not working" (= −1)
*edu*1/2	mother's educational status with categories "complete primary but incomplete secondary" (*edu*1 = 1), "complete secondary or higher" (*edu*2 = 1) and "no education or incomplete primary" (*edu*1 = *edu*2 = −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 1.1: Variables in the undernutrition data set.

1.3 Getting started

After having started the graphical user interface version of *BayesX*, a main window with four subwindows appears on the screen. These are the *command window* for entering and executing code, the *output window* for displaying results, the *review window* for easy access to past commands, and the *object browser* that displays all objects currently available. In the command line version of *BayesX*, there are, of course, no sub-windows but only command line prompt to enter commands.

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

> *objecttype objectname*

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

The rest of the tutorial is separated in seven parts dealing with the different steps of estimating a regression model with MCMC simulation techniques. In section 1.4, we create a *dataset object* to store, 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 1.5 describes how to create and handle these objects. Estimation of the regression model is carried out in section

1.6 using a *bayesreg object*. The next two sections describe how to visualise the estimation results and how to customise the obtained graphics. Section 1.9 describes post estimation commands which can be used to investigate the sampling paths and the autocorrelation functions of the estimated parameters. In a last section we perform a sensitivity analysis to assess the impact of hyperparameter choices on our estimation results.

If you have not done so yet, please download the data set and the *boundary file* associated with this tutorial now (from the *BayesX* homepage). 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.

1.4 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
```

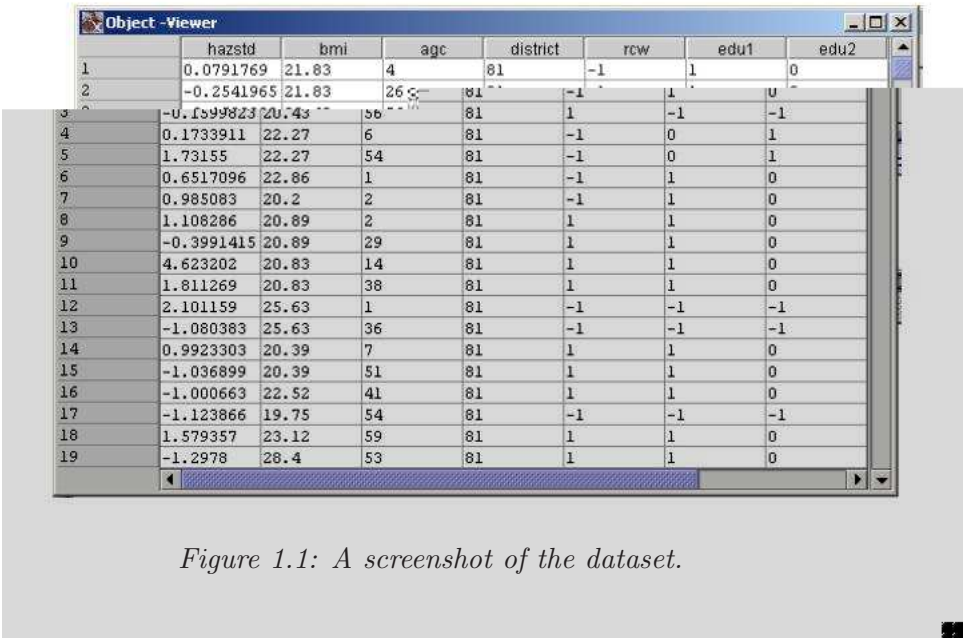
Option `maxobs` can be used to speed up the execution time of the `infile` command. If `maxobs` is specified, *BayesX* allocates enough memory to store all the data while the total amount of required memory is unknown in advance if `maxobs` remains unspecified. For larger data sets, this may cause *BayesX* 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 be 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 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 1.1). The same can also be achieved by double-clicking on the *dataset object* in the *object browser*.



	hazstd	bmi	agc	district	rcw	edu1	edu2
1	0.0791769	21.83	4	81	-1	1	0
2	-0.2541965	21.83	26	81	-1	1	0
3	-0.1999823	20.43	56	81	1	-1	-1
4	0.1733911	22.27	6	81	-1	0	1
5	1.73155	22.27	54	81	-1	0	1
6	0.6517096	22.86	1	81	-1	1	0
7	0.985083	20.2	2	81	-1	1	0
8	1.108286	20.89	2	81	1	1	0
9	-0.3991415	20.89	29	81	1	1	0
10	4.623202	20.83	14	81	1	1	0
11	1.811269	20.83	38	81	1	1	0
12	2.101159	25.63	1	81	-1	-1	-1
13	-1.080383	25.63	36	81	-1	-1	-1
14	0.9923303	20.39	7	81	1	1	0
15	-1.036899	20.39	51	81	1	1	0
16	-1.000663	22.52	41	81	1	1	0
17	-1.123866	19.75	54	81	-1	-1	-1
18	1.579357	23.12	59	81	1	1	0
19	-1.2978	28.4	53	81	1	1	0

Figure 1.1: A screenshot of the dataset.

Further methods allow to examine the variables in the *dataset object*. For a categorial variable such as *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
```

1.5 Map objects

In the following, we will 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 the boundaries using the `infile` command of *map objects*:

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

Having read 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 1.2.

```
        ⋮
"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

        ⋮
```

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 reference manual for a detailed description of some special cases, e.g. regions divided into subregions.

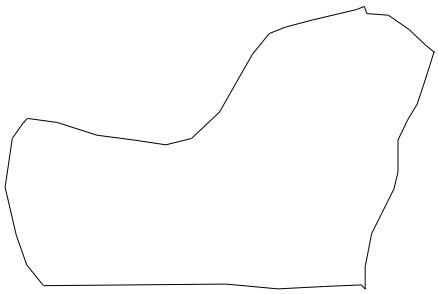


Figure 1.2: Corresponding graph of the section of the boundary file

Map objects may be visualised using method `describe`:

```
> m.describe
```

resulting in the graph shown in Figure 1.3. 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
```

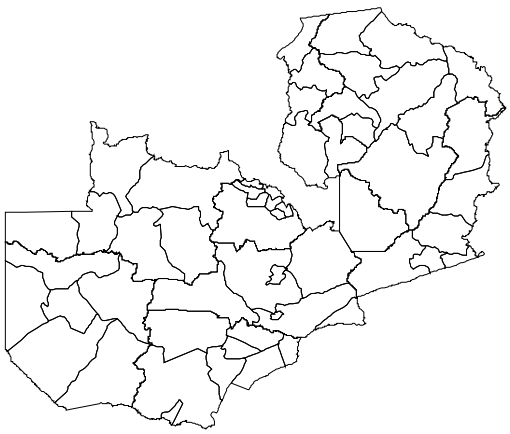


Figure 1.3: The districts within Zambia.

The numerical complexity associated with the estimation of structured spatial effects using MCMC techniques depends essentially on the structure of the neighbourhood matrix. Often the geographical information stored in a boundary file does not represent the "ideal" ordering of the districts or

regions (with respect to the estimation problem). Therefore it may be useful to reorder the map using method `reorder`:

```
> m.reorder
```

Usually, reordering results in a smaller bandwidth although the bandwidth is not the criterion that is minimised by `reorder`. Instead the *envelope* of the neighbourhood matrix is minimised (compare George & Liu (1981)).

In order to avoid reordering the *map object* every time you start *BayesX*, it is useful to store the reordered version in a separate file. This can be achieved using the `outfile` command of *map objects*:

```
> m.outfile, replace using c:\data\zambiasort.bnd
```

The reordered map is now stored in the given file. 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.

Reading the boundary information from an external file and computing the neighbourhood matrix may be a computationally intensive task if the map contains a large number of regions or if the polygons are given in great detail. To avoid doing these computation in every *BayesX* session, we store the neighbourhood information in a *graph file* using method `outfile` together with the `graph` option:

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

A graph file stores the nodes and the edges of a graph $G = (N, E)$, see for example George & Liu (1981) 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 typically will be the name of the 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 reordered map of Zambia:

```
57
87
1
5
76
3
9 8 7
67
2
10 9
```

$\vdots$

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 '87', which is the first region in the reordered map of Zambia. Region '87' has 1 neighbour, namely the 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 '76' and its three neighbours and lines 8 to 10 correspond to node '67'.

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
87
1
5 1.44172
76
3
7 8 9 0.707424 1.3816 0.682372
67
2
9 10 1.67424 0.8406
```

⋮

Here the edge of the first node '87' has weight 1.44172, the edges of the second node have weights 0.707424, 1.3816 and 0.682372.

Note, that graph files allow the estimation of very general correlated effects based on 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 three-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\zambiasort.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
```

1.6 Bayesian semiparametric regression

To estimate a regression model using MCMC simulation techniques, we first create a *bayesreg object*:

```
> bayesreg b
```

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 *bayesreg 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 *bayesreg objects*:

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

Note, that `outfile` does not only specify a directory but also a base filename (the character 'b' 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 'b'. Of course, the base filename may be different from the name of the *bayesreg object*.

In addition to parameter estimates, *BayesX* also gives acceptance rates for the different effects and 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\logmcmc.txt
```

After opening a *log file*, every information written to the output window is also stored in this 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:

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

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 $f^{\mathsf{str}}(district)$ and an uncorrelated part $f^{\mathsf{unstr}}(district)$, see Fahrmeir & 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 *bayesreg objects*:

```
> b.regress hazstd = rcw + edu1 + edu2 + tpr + sex + bmi(psplinerw2)
  + agc(psplinerw2) + district(spatial,map=m) + district(random),
  family=gaussian iterations=12000 burnin=2000 step=10 predict using d
```

Options `iterations`, `burnin` and `step` define properties of the MCMC algorithm. The total number of MCMC iterations is given by `iterations` while the number of burn in iterations is given by `burnin`. Therefore we obtain a sample of 10000 random numbers with the above specifications. Since, in general, these random numbers are correlated, we do not use all of them but thin out the Markov chain by the thinning parameter `step`. Specifying `step=10` as above forces *BayesX* to store only every 10th sampled parameter which leads to a random sample of length 1000 for every parameter in our example. Note, that the choice of `iterations` of course also affects computation time.

If option `predict` is specified, samples of the deviance, the effective number of parameters p_D, and the deviance information criteria DIC of the model are computed, see Spiegelhalter et al.

(2002). In addition, estimates for the linear predictor and the expectation of every observation are obtained.

In the following, we reproduce the content of the *output window* to make the user familiar with the estimation results produced by *BayesX*. Note that the output may look somewhat different depending on the version of *BayesX* you are considering.

```
ESTIMATION RESULTS:

  Predicted values:

  Estimated mean of predictors, expectation of response and
  individual deviances are stored in file
  c:\data\b_predictmean.raw

  Estimation results for the deviance:

  Unstandardized Deviance (-2*Loglikelihood(y|mu))

  Mean:             12688.726
  Std. Dev:         12.993345
  2.5% Quantile:    12663.772
  10% Quantile:     12672.535
  50% Quantile:     12688.173
  90% Quantile:     12705.219
  97.5% Quantile:   12714.784

  Saturated Deviance (-2*Loglikelihood(y|mu) + 2*Loglikelihood(y|mu=y))

  Mean:             4850.7212
  Std. Dev:         98.805525
  2.5% Quantile:    4658.3601
  10% Quantile:     4719.064
  50% Quantile:     4854.0429
  90% Quantile:     4981.005
  97.5% Quantile:   5046.1822

  Samples of the deviance are stored in file
  c:\data\b_deviance_sample.raw

  Estimation results for the DIC:

  DIC based on the unstandardized deviance

  Deviance(bar_mu):           12639.017
  pD:                         49.708864
  DIC:                        12738.435

  DIC based on the saturated deviance

  Deviance(bar_mu):           4800.0024
  pD:                         50.718841
  DIC:                        4901.44

  Estimation results for the scale parameter:

  Acceptance rate:   100 %

  Mean:             0.80205
  Std. dev.:        0.016386
  2.5% Quantile:    0.7711
```

```
10% Quantile:     0.780591
50% Quantile:     0.801434
90% Quantile:     0.824332
97.5% Quantile:   0.835445



FixedEffects1


Acceptance rate:    100 %


Variable  mean           Std. Dev.      2.5% quant.    median         97.5% quant.
const     0.102144       0.0468836      0.0115916      0.101498       0.196307
rcw       0.00780289     0.0136821      -0.0186219     0.00754262     0.0340403
edu1      -0.0622753     0.0265637      -0.116141      -0.0613026     -0.0125551
edu2      0.235591       0.0465318      0.142865       0.234172       0.325396
tpr       0.0898013      0.0216879      0.0483689      0.0901556      0.132811
sex       -0.0585167     0.0125313      -0.0826825     -0.0583656     -0.0339112

Results for fixed effects are also stored in file
c:\data\b_FixedEffects1.res


f_bmi_pspline


Acceptance rate:    100 %

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

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

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


f_bmi_pspline_variance


Acceptance rate:    100 %

Estimation results for the variance component:

Mean:             0.00174445
u44(d-52 (M4209)-3102(-0.780591) T 0-10.92TD (97.5%)-524(Quantate:)-310625(049780591) T 0-10.92TD (10%)-524(Quar
50% Quantile:
97.5% Quantile:

Results for the variance comffects are also stored in file
```

```
2.5% Quantile:    121.533            (df: 5.531)
10% Quantile:     211.716            (df: 4.85283)
50% Quantile:     717.565            (df: 3.58964)
90% Quantile:     1910.98            (df: 2.78449)
97.5% Quantile:   3038.87            (df: 2.46105)

Results for the smoothing parameter are also stored in file
c:\data\b_f_bmi_pspline_lambda.res


f_agc_pspline


Acceptance rate:    100 %

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

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

Results may be visualized using method 'plotnonp'
Type for example: objectname.plotnonp 3


f_agc_pspline_variance


Acceptance rate:    100 %

Estimation results for the variance component:

Mean:             0.00630648
Std. dev.:        0.00788173
2.5% Quantile:    0.00124029
10% Quantile:     0.00179477
50% Quantile:     0.0041396
90% Quantile:     0.0121621
97.5% Quantile:   0.0228385

Results for the variance component are also stored in file
c:\data\b_f_agc_pspline_var.res

Estimation results for the smoothing parameter:

Mean:             234.684            (df: 6.28883)
Std. dev.:        177.494
2.5% Quantile:    34.9805            (df: 9.37168)
10% Quantile:     66.326             (df: 8.23766)
50% Quantile:     194.159            (df: 6.55631)
90% Quantile:     442.421            (df: 5.457)
97.5% Quantile:   647.435            (df: 5.0029)

Results for the smoothing parameter are also stored in file
c:\data\b_f_agc_pspline_lambda.res


f_district_spatial


Acceptance rate:    100 %
```

```
Results are stored in file
c:\data\b_f_district_spatial.res

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

Results may be visualized in BayesX using method 'drawmap'
Type for example: objectname.drawmap 5



f_district_spatial_variance


Acceptance rate:    100 %

Estimation results for the variance component:

Mean:             0.0360104
Std. dev.:        0.0179351
2.5% Quantile:    0.0105888
10% Quantile:     0.0165585
50% Quantile:     0.0324881
90% Quantile:     0.0600347
97.5% Quantile:   0.0826229

Results for the variance component are also stored in file
c:\data\b_f_district_spatial_var.res

Estimation results for the smoothing parameter:

Mean:             28.7933            (df: 0)
Std. dev.:        16.8091
2.5% Quantile:    9.73429            (df: 0)
10% Quantile:     13.3299            (df: 0)
50% Quantile:     24.7758            (df: 0)
90% Quantile:     48.715             (df: 0)
97.5% Quantile:   75.0528            (df: 0)

Results for the smoothing parameter are also stored in file
c:\data\b_f_district_spatial_lambda.res


f_district_random


Acceptance rate:    100 %

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

Results for the sum of the structured and unstructured
spatial effects are stored in file
c:\data\b_district_spatialtotal.res


f_district_random_variance


Acceptance rate:    100 %

Estimation results for the variance component:
```

```
Mean:             0.00788548
Std. dev.:        0.00618319
2.5% Quantile:    0.000696144
10% Quantile:     0.00138013
50% Quantile:     0.00645178
90% Quantile:     0.0165836
97.5% Quantile:   0.0237649

Results for the variance component are also stored in file
c:\data\b_f_district_random_var.res

Files of model summary:

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

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

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

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

Batch file for visualizing effects of nonlinear functions
in R / S-Plus is stored in file
c:\data\b_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\b_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. The files contain the posterior mean and median, the posterior 2.5%, 10%, 90% and 97.5% quantiles, and the corresponding 95% and 80% posterior probabilities of the estimated effects. For example, the beginning of the file `c:\data\b_f_bmi_pspline.res` for the effect of *bmi* may look like this:

intnr	bmi	pmean	pqu2p5	pqu10	pmed	pqu90	pqu97p5	pcat95	pcat80
1	12.8	-0.284065	-0.660801	-0.51678	-0.283909	-0.0585753	0.085998	0	-1
2	13.15	-0.276772	-0.609989	-0.483848	-0.275156	-0.070517	0.0572406	0	-1
3	14.01	-0.258674	-0.515628	-0.416837	-0.257793	-0.10009	-0.00289024	-1	-1

The posterior quantiles and posterior probabilities may be changed by the user via 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 0.5%, 15%, 85% and 99.5% quantiles of the posterior. 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. For example, 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). In addition to the *ps files* a file containing the commands to reproduce the graphics is stored in the output directory. In our example the name of the file is `c:\data\b_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*.

1.7 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 1.8 describes how to customise plots.

1.7.1 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`:

```
> b.plotnonp 1
```

and

```
> b.plotnonp 3
```

produce the graphs shown in Figure 1.4 in an *object-viewer window*. The numbers following the `plotnonp` command depend on the order in which the model terms have been specified (and an internal ordering of the effect types). The numbers are supplied in the *output window* after estimation, compare the results in the previous section.

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

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

```
> b.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 mean. For the structured spatial effect this can be achieved using the post estimation command `drawmap`

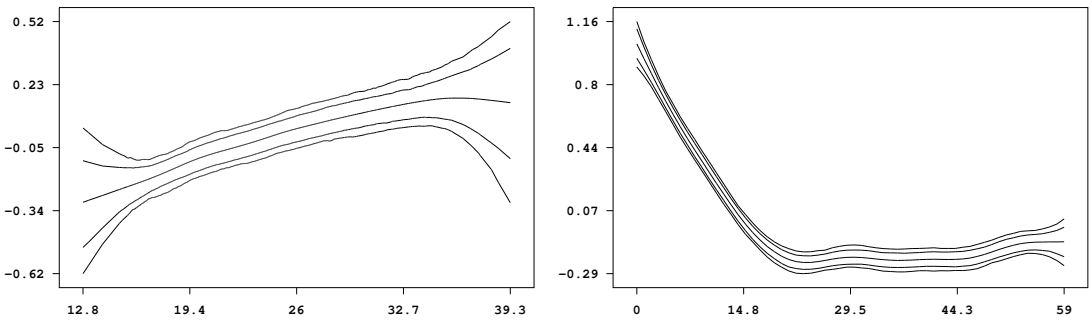


Figure 1.4: Effect of the body mass index of the child's mother and of the age of the child together with pointwise 80% and 95% credible intervals.

```
> b.drawmap 5
```

which results in the graph shown in Figure 1.5.

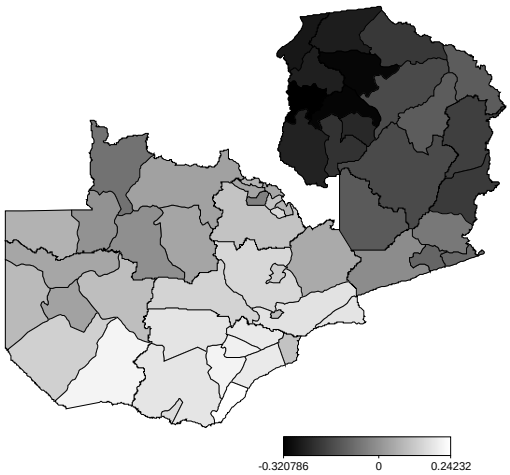


Figure 1.5: Posterior mean of the structured spatial effect.

1.7.2 Graph Objects

The commands presented in the previous subsection work only after having estimated a regression model in the current *BayesX* session. However, it may of course also be useful to visualise results of former analyses. 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.

In a first step, 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\b_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 pmean pqu2p5 pqu10 pqu90 pqu97p5 using res
```

reproduces the graph in Figure 1.4.

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\b_f_district_spatial.res
```

stores the estimation results for the structured spatial effect in the *dataset object* `res`. Now we can visualise the posterior mean using method `drawmap` of *graph objects* leading again to the graph shown in Figure 1.5:

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

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

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

```
> g.drawmap pcat95 district, map=m using res
```

The result is shown in Figure 1.6.

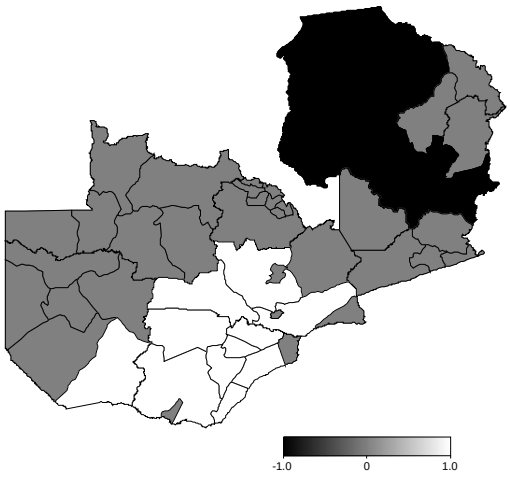


Figure 1.6: Posterior 95% probability of the structured spatial effect.

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 1.7:

```
> res.infile using c:\data\b_f_district_random.res
> g.drawmap pmean district, map=m using res
```

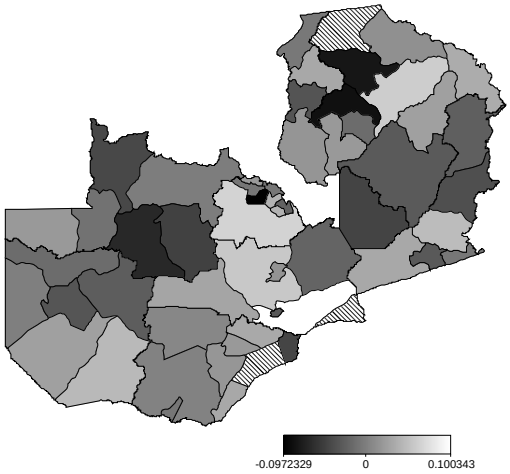


Figure 1.7: Posterior mean of the unstructured spatial effect.

1.8 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. Hence, 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 intervals 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 1.6). 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 1.8):

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

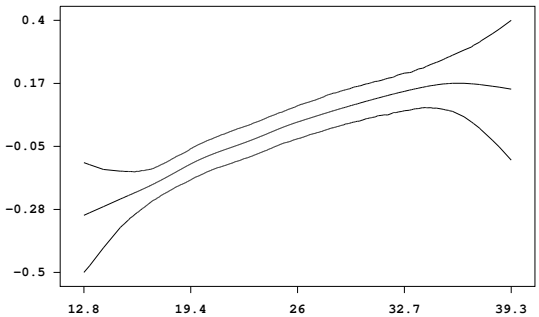


Figure 1.8: Effect of the body mass index of the child's mother with pointwise 80% credible intervals only.

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 1.9 for the resulting plots):

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

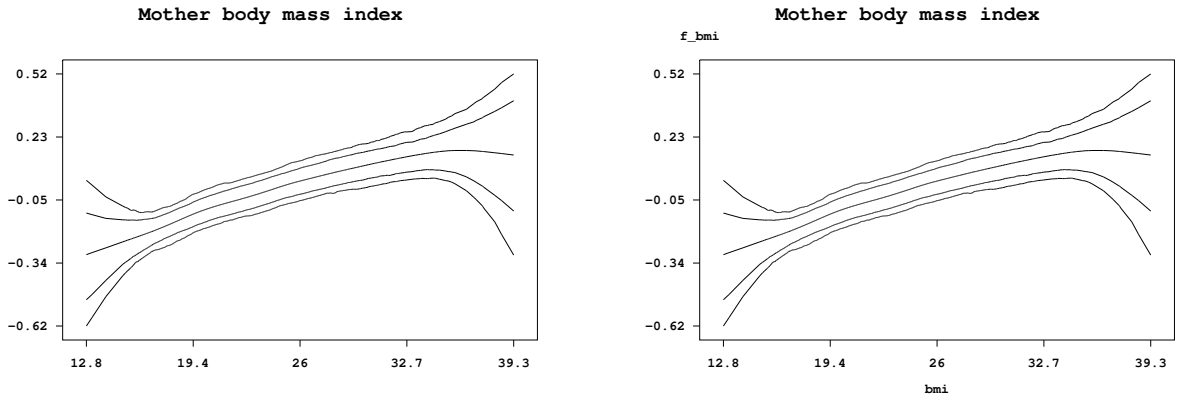


Figure 1.9: Specification of title and axis labels.

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 1.10.

```
> b.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 1.10 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*.

```
> b.plotnonp 3, xlab="age" ylab="f_age" title="Age of the child in months"
  ylimbottom=-0.3  ystep=0.3 xlimbottom=0 xlimtop=60 xstep=10
```

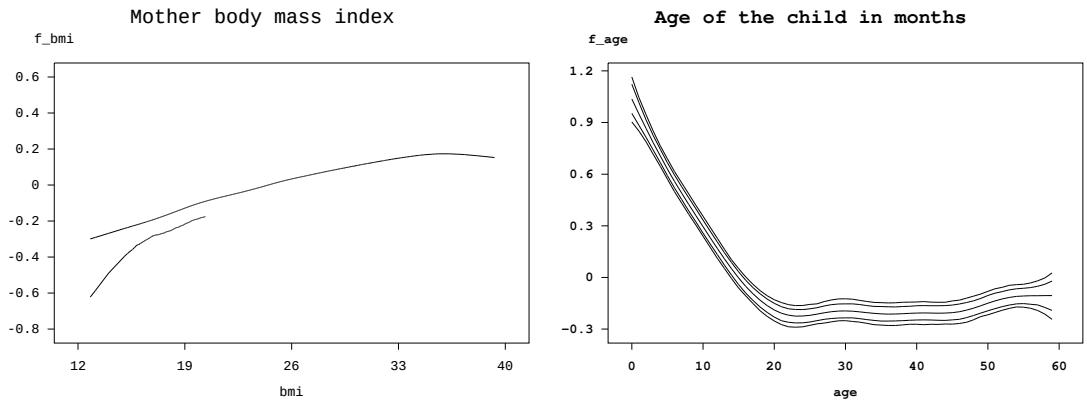


Figure 1.10: Re-defining x- and y-axis.

Now we turn to the options for method `drawmap`. By default, `drawmap` uses grey scales to represent different values of the posterior mean. 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

```
> b.drawmap 5, color swapcolors
```

leads to the graph shown in Figure 1.11 with higher values being represented through red colours and smaller values through green colours. An alternative color scheme can be requested by adding option `hcl`.

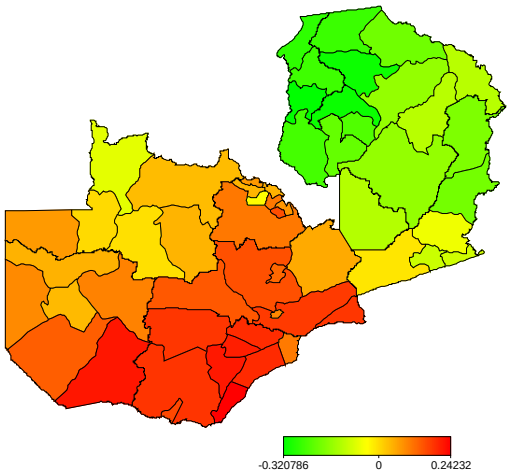


Figure 1.11: Posterior mean of the structured spatial effect in colour.

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`

```
> b.drawmap 5, color swapcolors title="Structured spatial effect"
```

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

```
> b.drawmap 5, color swapcolors title="Structured spatial effect" lowerlimit=-0.3
  upperlimit=0.3
```

The graph produced by the second command is shown in Figure 1.12.

1.9 Autocorrelation functions and sampling paths

Bayesreg objects provide some post estimation commands to get sampled parameters or to plot autocorrelation functions of sampled parameters. For example

```
> b.plotautocor, maxlag=250
```

computes and displays the autocorrelation functions for all estimated parameters with `maxlag` specifying the maximum lag number (Figure 1.13 shows a small part of the resulting graph).

If the number of parameters is large, this may be computationally expensive, so *BayesX* provides a second possibility to compute autocorrelation functions. Adding the option `mean` to the `plotautocor` command as in

```
> b.plotautocor, mean
```

leads to the computation of only the minimum, mean and maximum autocorrelation functions. The result for the scale parameter is shown in Figure 1.14.

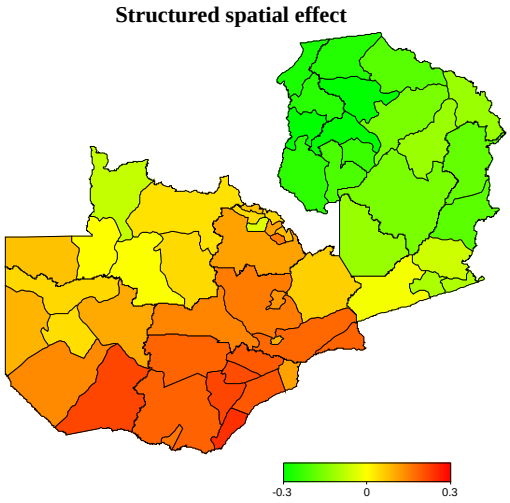


Figure 1.12: Specifying a title and the range of the plot for spatial effects.

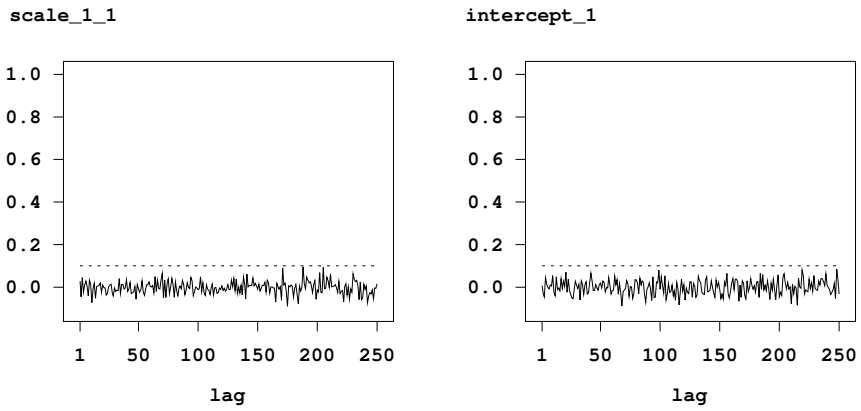


Figure 1.13: Autocorrelation function for the scale parameter and the intercept.

Note, that executing the `plotautocor` command also stores the computed autocorrelation functions in a file named `autocor.raw` in the output directory of the *bayesreg object*. The content of this file can then be visualised in your favorite statistical software package.

To save memory, the sampling paths of the estimated parameters are only stored temporarily by default and will be destroyed, when the corresponding *bayesreg object* is deleted. If we want to store the sampling paths permanently, we have to execute the `getsample` command

```
> b.getsample
```

which stores the sampled parameters in ASCII files in the output directory. To avoid too large files, the samples are typically partitioned into several files. Executing the `getsample` command also produces *ps files* of the sampling paths in the output directory (compare Figure 1.15 for the content of one of these files).

1.10 Sensitivity analysis

In some situations, the estimation results of a full Bayesian semiparametric regression model depend on the choice of hyperparameters, e.g. the parameters a and b defining the inverse gamma prior of

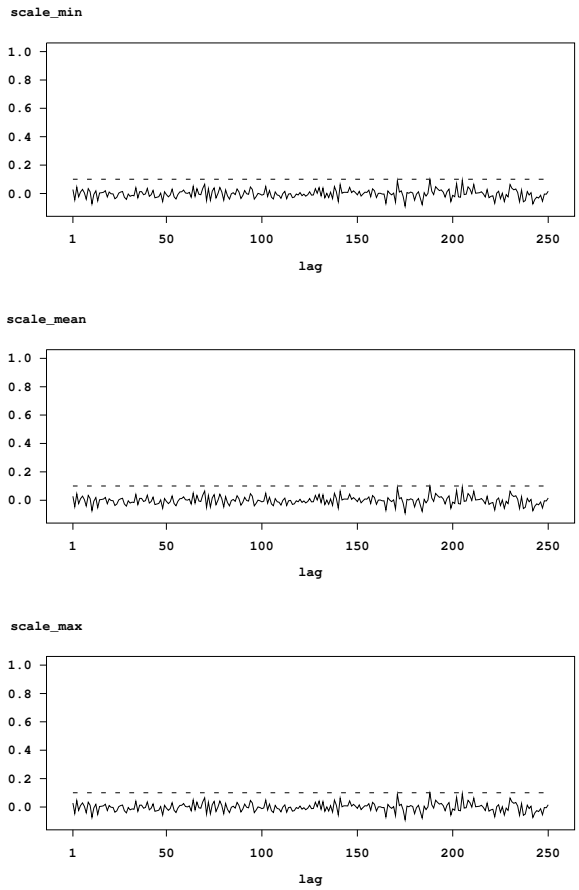


Figure 1.14: Minimum, mean and maximum autocorrelation function for the scale parameter.

the variances of nonparametric and spatial effects. Hence, it is often recommended to check how sensitive the results are with respect to changes in the hyperparameters. In the following we will re-estimate the model from section 1.6 with different choices for the hyperparameters a and b for each effect in the model. The standard choices for a and b are $a = b = 0.001$. As a first trial we choose a smaller value for a and b:

```
> b.regress hazstd = rcw + edu1 + edu2 + tpr + sex
  + bmi(psplinerw2,a=0.00001,b=0.00001) + agc(psplinerw2,a=0.00001,b=0.00001)
  + district(spatial,map=m,a=0.00001,b=0.00001)
  + district(random,a=0.00001,b=0.00001), family=gaussian iterations=12000
    burnin=2000 step=10 predict using d
```

Figure 1.16 shows the results for the nonparametric effects with this choice of hyperparameters. Obviously, the estimated functions are somewhat smoother but they do not differ that much from the estimates with the standard choices.

Now we try two further choices for the hyperparameters, each with $a = 1$ and b small. We estimate models with $b = 0.005$ and $b = 0.00005$:

```
> b.regress hazstd = rcw + edu1 + edu2 + tpr + sex + bmi(psplinerw2,a=1,b=0.005)
  + agc(psplinerw2,a=1,b=0.005) + district(spatial,map=m,a=1,b=0.005)
  + district(random,a=1,b=0.005), family=gaussian iterations=12000 burnin=2000
  step=10 predict using d
```

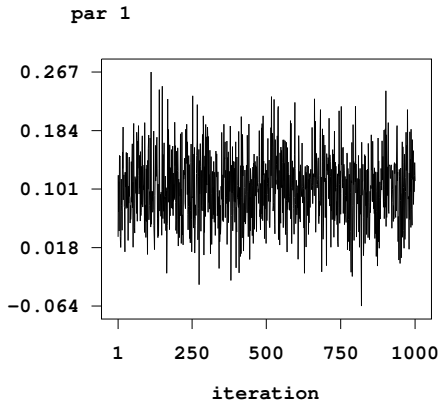


Figure 1.15: Sampling path of the intercept.

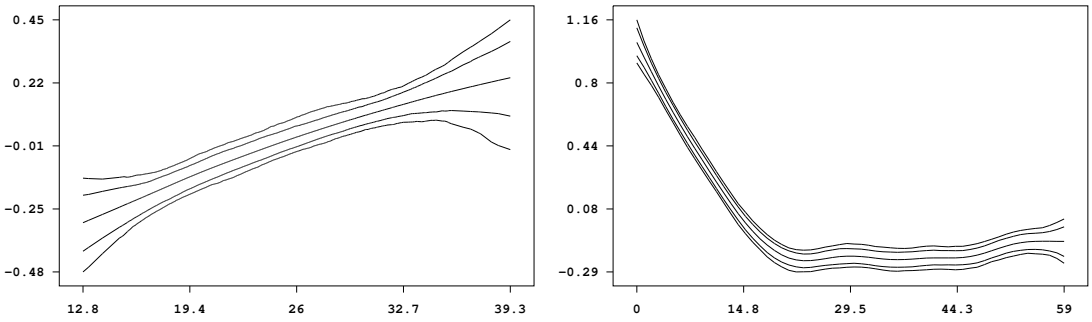


Figure 1.16: Results for the nonparametric effects with hyperparameters $a = b = 0.00001$ *for nonparametric and spatial effects.*

```
> b.regress hazstd = rcw + edu1 + edu2 + tpr + sex + bmi(psplinerw2,a=1,b=0.00005)
  + agc(psplinerw2,a=1,b=0.00005) + district(spatial,map=m,a=1,b=0.00005)
  + district(random,a=1,b=0.00005), family=gaussian iterations=12000 burnin=2000
  step=10 predict using d
```

Figure 1.17 and 1.18 contain the results for the nonparametric effects for the two choices of hyperparameters.

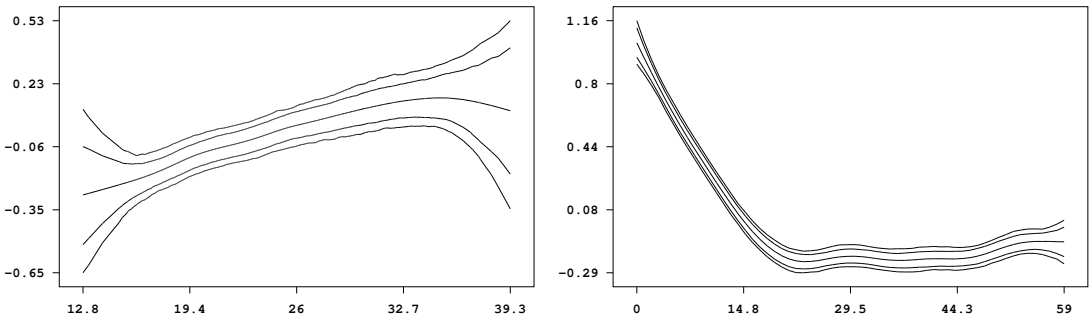


Figure 1.17: Results for the nonparametric effects with hyperparameters $a = 1$ *and* $b = 0.005$ *for nonparametric and spatial effects.*

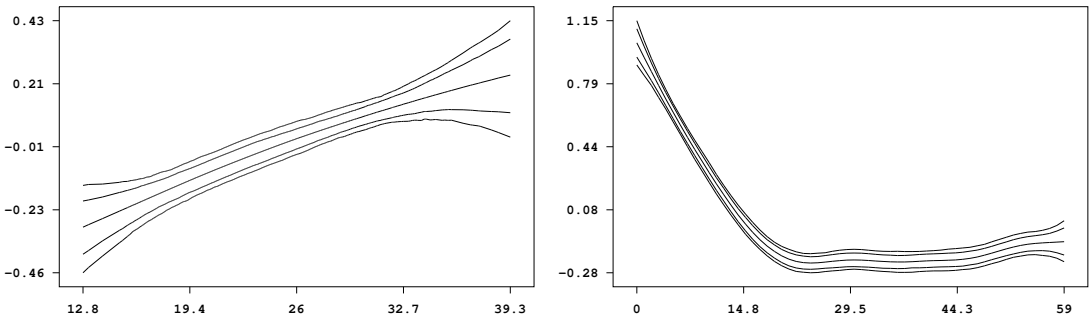


Figure 1.18: Results for the nonparametric effects with hyperparameters $a = 1$ *and* $b = 0.00005$ *for nonparametric and spatial effects.*

Tutorial 2

Determinants of childhood undernutrition in Zambia: A tutorial on Bayesian semiparametric regression using GLMM methodology

2.1 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 childhood undernutrition 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 obtain them.

The main focus in this tutorial is on empirical Bayes inference based on mixed model methodology. *BayesX* also supports two further inferential concepts: full Bayesian inference based on MCMC and a method for simultaneously selecting variables and smoothing parameters, which are described in two additional tutorials. 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 2.3–2.5.

The theoretical background of Bayesian semiparametric regression will not be described in this tutorial. The reference manual may serve as a first introduction, while further details about the estimation techniques for the empirical Bayes approach in the context of univariate exponential families can be found in Fahrmeir, Kneib & Lang (2004). Categorical extensions are treated in Kneib & Fahrmeir (2006) while Kneib (2006) and Kneib & Fahrmeir (2007) deal with the analysis of continuous time survival analysis.

2.2 Description of the data set

Undernutrition among children is usually determined by assessing an anthropometric status of the children 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 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), while MAI and σ correspond to the median and the standard deviation in the reference population, respectively.

Our 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 2.1 gives a description of the variables that we will use in our model.

Variable	Description
hazstd	standardised Z-score for stunting
bmi	body mass index of the mother
agc	age of the child in months
district	district where the mother lives
rcw	mother's employment status with categories "working" (= 1) and "not working" (= −1)
*edu*1/2	mother's educational status with categories "complete primary but incomplete secondary" (*edu*1 = 1), "complete secondary or higher" (*edu*2 = 1) and "no education or incomplete primary" (*edu*1 = *edu*2 = −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 2.1: Variables in the undernutrition data set.

2.3 Getting started

After having started the graphical user interface version of *BayesX*, a main window with four subwindows appears on the screen. These are the *command window* for entering and executing code, the *output window* for displaying results, the *review window* for easy access to past commands, and the *object browser* that displays all objects currently available. In the command line version of *BayesX*, there are, of course, no sub-windows but only command line prompt to enter commands.

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

> *objecttype objectname*

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

The rest of the tutorial is separated in five parts dealing with the different steps of estimating a regression model based on mixed model methodology. In section 2.4, we create a *dataset object* to store, 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 2.5 describes how to create and handle these objects. Estimation of the regression model is carried out in section 2.6

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 (from the *BayesX* homepage). 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.

2.4 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.

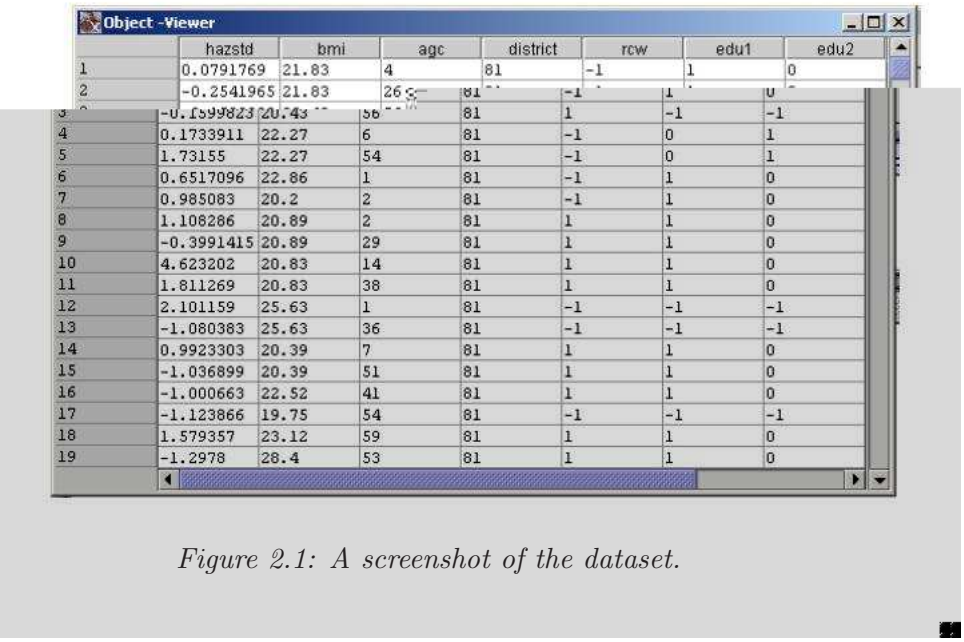
A second option that may be added to the `infile` command is the `missing` option to indicate missing values. Specifying for example `missing = M` defines the letter '`M`' as an indicator for a missing value. The default for missing values are a period '.' and '`NA`' (which remain valid indicators for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

After having read 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 2.1). The same can also be achieved by double-clicking on the *dataset object* in the *object browser*.

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



	hazstd	bmi	agc	district	rcw	edu1	edu2
1	0.0791769	21.83	4	81	-1	1	0
2	-0.2541965	21.83	26	81	-1	1	0
3	-0.1999823	20.43	56	81	1	-1	-1
4	0.1733911	22.27	6	81	-1	0	1
5	1.73155	22.27	54	81	-1	0	1
6	0.6517096	22.86	1	81	-1	1	0
7	0.985083	20.2	2	81	-1	1	0
8	1.108286	20.89	2	81	1	1	0
9	-0.3991415	20.89	29	81	1	1	0
10	4.623202	20.83	14	81	1	1	0
11	1.811269	20.83	38	81	1	1	0
12	2.101159	25.63	1	81	-1	-1	-1
13	-1.080383	25.63	36	81	-1	-1	-1
14	0.9923303	20.39	7	81	1	1	0
15	-1.036899	20.39	51	81	1	1	0
16	-1.000663	22.52	41	81	1	1	0
17	-1.123866	19.75	54	81	-1	-1	-1
18	1.579357	23.12	59	81	1	1	0
19	-1.2978	28.4	53	81	1	1	0

Figure 2.1: A screenshot of the dataset.

```
> 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
```

2.5 Map objects

In the following, we will 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 the boundaries using the `infile` command of *map objects*:

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

Having read 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 2.2.

⋮

```
"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
```

⋮

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 reference manual for a detailed description of some special cases, e.g. regions divided into subregions.

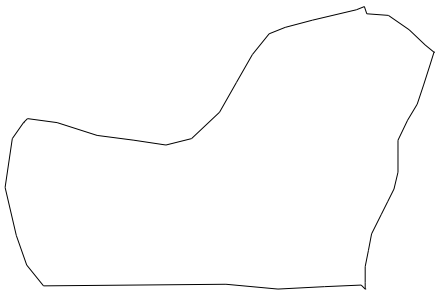


Figure 2.2: Corresponding graph of the section of the boundary file

Map objects may be visualised using method `describe`:

```
> m.describe
```

resulting in the graph shown in Figure 2.3. 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
```

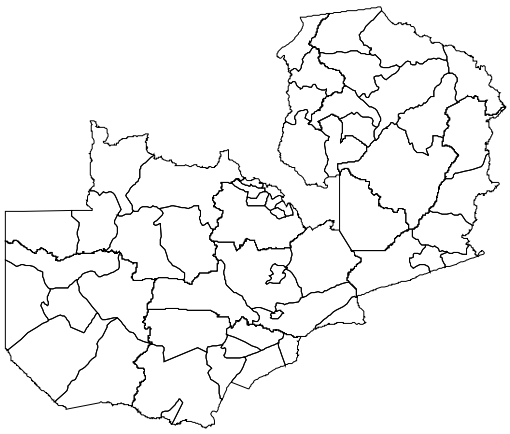


Figure 2.3: The districts within Zambia.

Reading the boundary information from an external file and computing the neighbourhood matrix may be a computationally intensive task if the map contains a large number of regions or if the polygons are given in great detail. To avoid doing these computation in every *BayesX* session, we store the neighbourhood information in a *graph file* using method `outfile` together with the `graph` option:

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

A graph file stores the nodes and the edges of a graph $G = (N, E)$, see for example George & Liu (1981) 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 typically will be the name of the 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 reordered map of Zambia:

```
57
87
1
5
76
3
9 8 7
67
2
10 9
```

⋮

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 '87', which is the first region in the reordered map of Zambia. Region '87' has 1 neighbour, namely the 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 '76' and its three neighbours and lines 8 to 10 correspond to node '67'.

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
87
1
5 1.44172
76
3
7 8 9 0.707424 1.3816 0.682372
67
2
9 10 1.67424 0.8406
```

⋮

Here the edge of the first node '87' has weight 1.44172, the edges of the second node have weights 0.707424, 1.3816 and 0.682372.

Note, that graph files allow the estimation of very general correlated effects based on 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 three-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\zambiasort.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
```

2.6 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 file names will 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 this 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:

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

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 $f^{\mathsf{str}}(district)$ and an uncorrelated part $f^{\mathsf{unstr}}(district)$, see Fahrmeir & 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 the reference 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.

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*. Note that the output may look somewhat different depending on the version of *BayesX* you are considering.

```
ESTIMATION RESULTS:


  Estimated scale parameter: 0.802145

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


  f_bmi_pspline


  Estimated variance:  1.14819e-05
  Inverse variance:    87093.8
  Smoothing parameter: 69861.9
  (Smoothing parameter = scale / variance)
  Degrees of freedom: 1.17481
  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: 249.001
(Smoothing parameter = scale / variance)
Degrees of freedom: 6.19468

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:  0.0294011
Inverse variance:    34.0123
Smoothing parameter: 27.2828
(Smoothing parameter = scale / variance)
Degrees of freedom: 16.7629

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:  0.00806667
Inverse variance:    123.967
```

```
Smoothing parameter: 99.4393
(Smoothing parameter = scale / variance)
Degrees of freedom: 14.2076

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     0.0610359      0.0341574      0.0734909      -0.00592615    0.127998
rcw       0.00767163     0.0136563      0.573859       -0.0191002     0.0344435
edu1      -0.0605106     0.0261369      0.020636       -0.111749      -0.00927198
edu2      0.234918       0.0459925      8.82492e-06    0.144754       0.325081
tpr       0.0904094      0.0218891      0.000123529    0.047498       0.133321
sex       -0.0585716     0.0129304      4.08485e-05    -0.0839203     -0.0332229

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


Model Fit


-2*log-likelihood:                3733.37
Degrees of freedom:               44.34
(conditional) AIC:                3822.05
(conditional) BIC:                4109.65
GCV:                              0.809435

Results on the model fit are stored in file
c:\data\r_modelfit.raw


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 R / S-Plus is stored in file
c:\data\r_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 and degrees of freedom. 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 (unless other levels have been requested). For example, the beginning of the file `c:\data\r_f_bmi_pspline.res` for the effect of *bmi* may look 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 via 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. For example, 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*.

2.7 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 *remlreg 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 2.8 describes how to customise plots.

2.7.1 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 2.4 in an *object-viewer window*. The numbers following the `plotnonp` command depend on the order in which the model terms have been specified (and an internal ordering of the effect types). 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. Hence, by default the plots include pointwise 80% and 95% credible intervals.

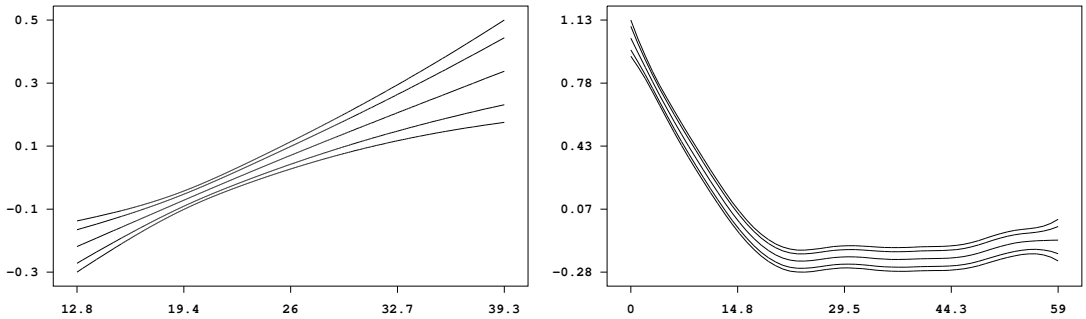


Figure 2.4: Effect of the body mass index of the child's mother and of the age of the child together with pointwise 80% and 95% credible intervals.

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 2.5.

2.7.2 Graph Objects

The commands presented in the previous subsection work only after having estimated a regression model in the current *BayesX* session. However, it may of course also be useful to visualise results

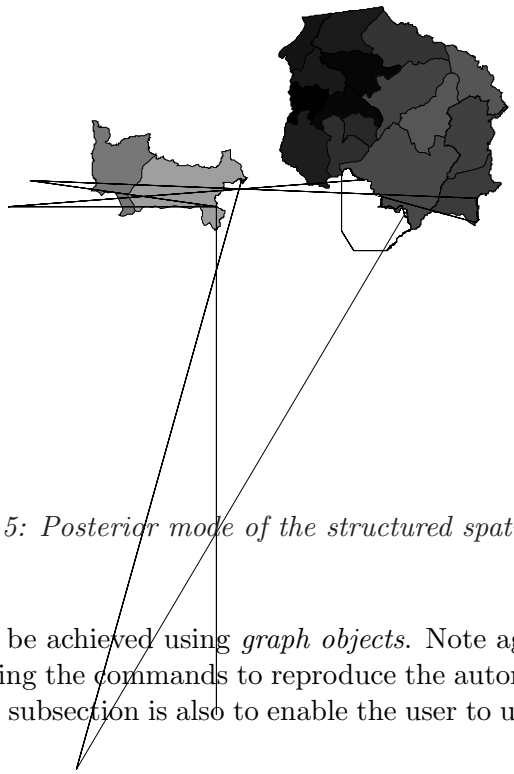


Figure 2.5: Posterior mode of the structured spatial effect.

of former analyses. 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.

In a first step, 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 2.4.

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 2.5:

```
> 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 2.6.

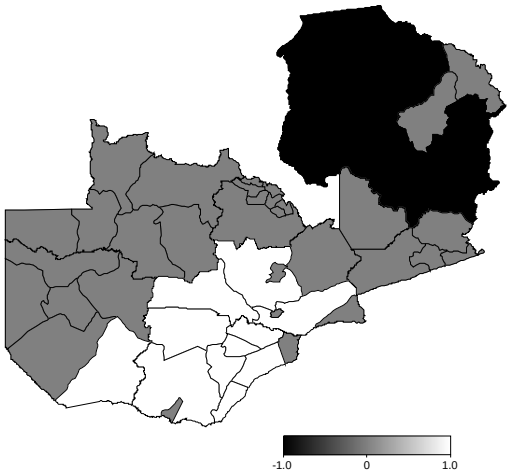


Figure 2.6: Posterior 95% probability of the structured spatial effect.

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 2.7:

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

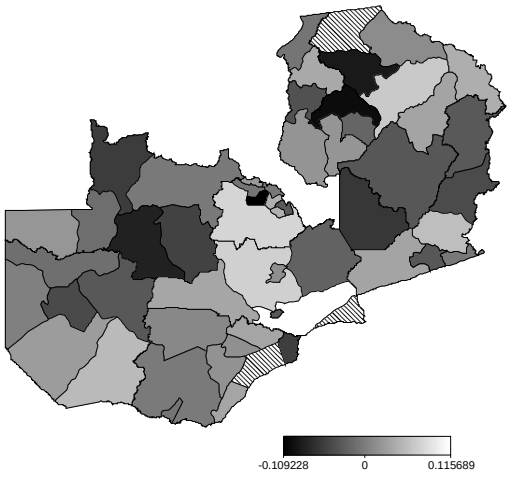


Figure 2.7: Posterior mode of the unstructured spatial effect.

2.8 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. Hence,

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 intervals 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 2.6). If the default values of `level1` and `level2` have been used, specifying `level=2` in the `plotnonp`

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 2.10.

```
> 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 2.10 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*.

```
> 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
```

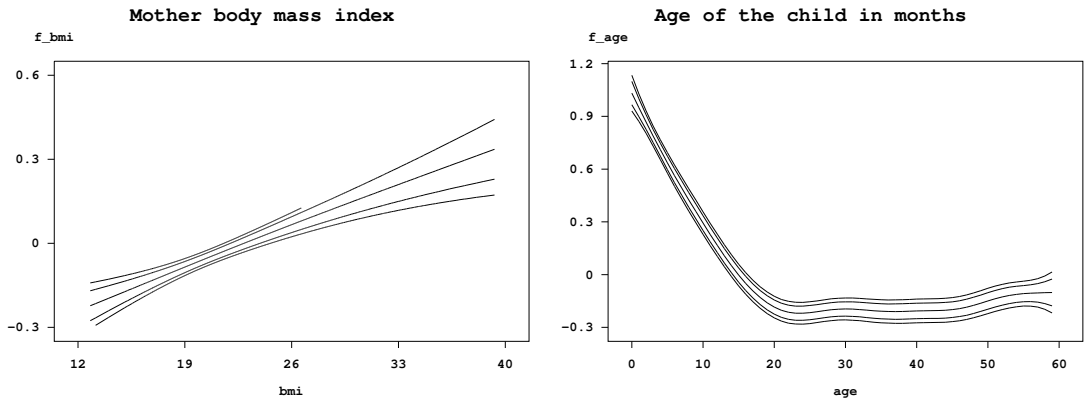


Figure 2.10: Re-defining x- and y-axis.

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 2.11 with higher values being represented through red colours and smaller values through green colours. An alternative color scheme can be requested by adding option `hcl`.

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
```

The graph produced by the second command is shown in Figure 2.12.

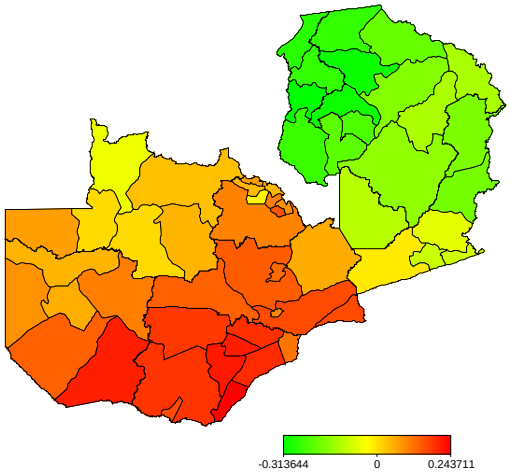


Figure 2.11: Posterior mode of the structured spatial effect in colour.

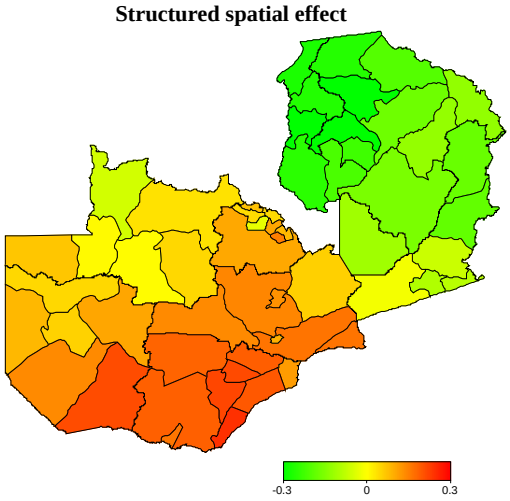


Figure 2.12: Specifying a title and the range of the plot for spatial effects.

Tutorial 3

Determinants of childhood undernutrition in Zambia: A tutorial on penalized least squares estimation including built in model and variable selection in semiparametric regression models

3.1 Introduction

This tutorial demonstrates the usage of *BayesX* for simultaneously selecting variables and smoothing parameters in semiparametric regression models using *stepwisereg objects*. As an example, we consider data on childhood undernutrition in Zambia. This data has already been analysed in Kandala et al. (2001) and we will use the same data set that has been used 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 obtain them.

The main focus in this tutorial is on a method for simultaneously selecting variables and smoothing parameters. *BayesX* also supports two further inferential concepts: full Bayesian inference based on MCMC and empirical Bayes inference based on mixed model methodology, which are described in two additional tutorials. 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 3.3–3.5.

The theoretical background of Bayesian semiparametric regression will not be described in this tutorial. The reference manual may serve as a first introduction, while further details about the estimation techniques employed in this tutorial can be found in Belitz & Lang (2008).

3.2 Description of the data set

Undernutrition among children is usually determined by assessing an anthropometric status of the children 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 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), while MAI and σ correspond to the median and the standard deviation in the reference population, respectively.

Our main interest is in 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 3.1 gives a description of the variables that will be used in our analysis.

Variable	Description
hazstd	standardised Z-score for stunting
bmi	body mass index of the mother
agc	age of the child in months
district	district where the mother lives
rcw	mother's employment status with categories "working" (= 1) and "not working" (= −1)
*edu*1/2	mother's educational status with categories "complete primary but incomplete secondary" (*edu*1 = 1), "complete secondary or higher" (*edu*2 = 1) and "no education or incomplete primary" (*edu*1 = *edu*2 = −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)
edu	mother's educational status with categories "no education or incomplete primary" (*edu* = 0), "complete primary but incomplete secondary" (*edu* = 1) and "complete secondary or higher" (*edu* = 2)

Table 3.1: Variables in the undernutrition data set.

3.3 Getting started

After having started the graphical user interface version of *BayesX*, a main window with four sub-windows appears on the screen. These are the *command window* for entering and executing code, the *output window* for displaying results, the *review window* for easy access to past commands, and the *object browser* that displays all objects currently available. In the command line version of *BayesX*, there are, of course, no sub-windows but only the command line prompt to enter commands.

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

```
> objecttype objectname
```

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

The rest of the tutorial is separated in seven parts dealing with the different steps of estimating a regression model. In section 3.4, we create a *dataset object* to store, 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 neighborhood information of the map of Zambia. This information will be stored in a *map object*. 3.5 describes how to create and handle these objects. Model selection and estimation of the regression model is carried out in 3.6 using a *stepwisereg object*. Section 3.7 extends the usage of *stepwisereg objects* to calculating credible bands. The next two sections describe how to visualize the estimation results and how to customize the obtained graphics.

If you have not done so yet, please download the data set and the *boundary file* associated with this tutorial now (from the *BayesX* homepage). 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.

3.4 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. It is therefore not necessary to specify the variable names explicitly 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 be 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 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 3.1). The

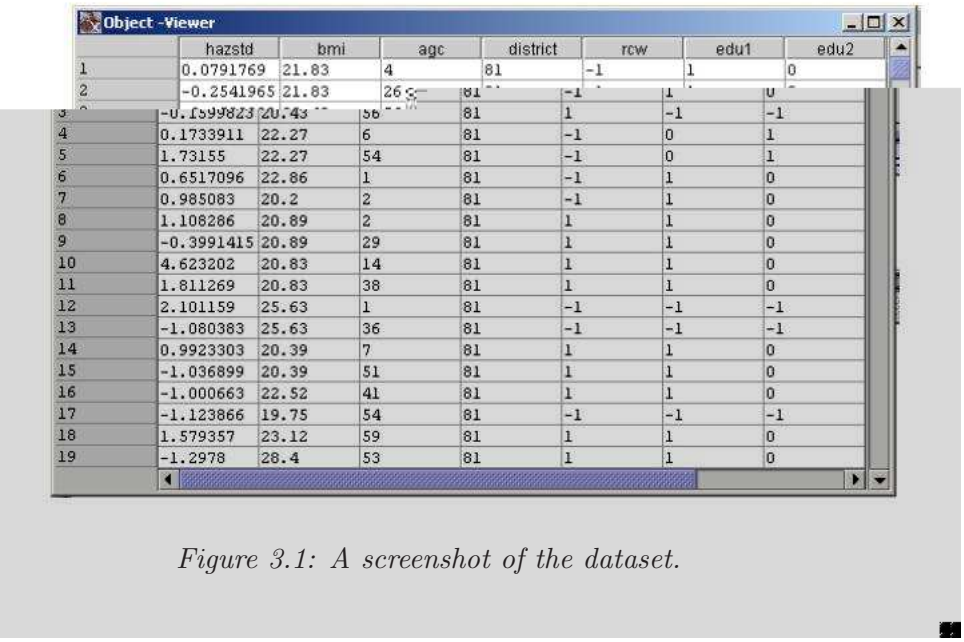


Figure 3.1: A screenshot of the dataset.

same can also be achieved by double-clicking on the *dataset object* in the *object browser*.

Further methods allow to examine the variables in the *dataset object*. For a categorial variable such as *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
```

3.5 Map objects

In the following, we will 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 the boundaries using the `infile` command of *map objects*:

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

Having read 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 3.2.

```
        ⋮
"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

⋮

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 reference manual for a detailed description of some special cases, e.g. regions divided into subregions.

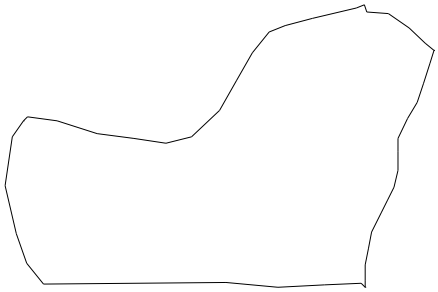


Figure 3.2: Corresponding graph of the section of the boundary file

Map objects may be visualised using method `describe`:

```
> m.describe
```

resulting in the graph shown in Figure 3.3. 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
```

The numerical complexity associated with the estimation of structured spatial effects depends essentially on the structure of the neighbourhood matrix. Often the geographical information stored in a boundary file does not represent the "ideal" ordering of the districts or regions (with respect to the estimation problem). Therefore it may be useful to reorder the map using method `reorder`:

```
> m.reorder
```

Usually, reordering results in a smaller bandwidth although the bandwidth is not the criterion that is minimised by `reorder`. Instead the *envelope* of the neighbourhood matrix is minimised (compare George & Liu (1981)).

In order to avoid reordering the *map object* every time you start *BayesX*, it is useful to store the reordered version in a separate file. This can be achieved using the `outfile` command of *map objects*:

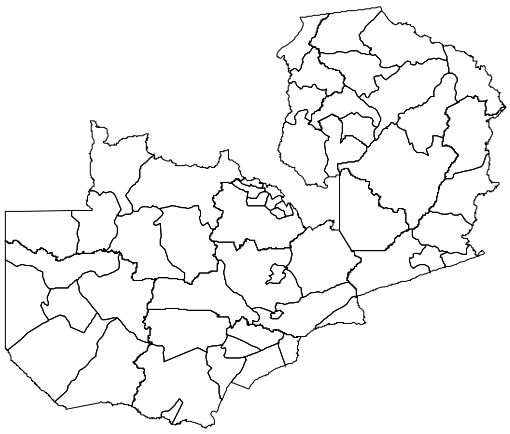


Figure 3.3: The districts within Zambia.

```
> m.outfile, replace using c:\data\zambiasort.bnd
```

The reordered map is now stored in the given file. 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.

Reading the boundary information from an external file and computing the neighbourhood matrix may be a computationally intensive task if the map contains a large number of regions or if the polygons are given in great detail. To avoid doing these computation in every *BayesX* session, we store the neighbourhood information in a *graph file* using method `outfile` together with the `graph` option:

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

A graph file stores the nodes and the edges of a graph $G = (N, E)$, see for example George & Liu (1981) 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 typically will be the name of the 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 reordered map of Zambia:

```
57
87
1
5
76
3
9 8 7
```

```
67
2
10 9
```

⋮

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 '87', which is the first region in the reordered map of Zambia. Region '87' has 1 neighbour, namely the 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 '76' and its three neighbours and lines 8 to 10 correspond to node '67'.

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
87
1
5 1.44172
76
3
7 8 9 0.707424 1.3816 0.682372
67
2
9 10 1.67424 0.8406
```

⋮

Here the edge of the first node '87' has weight 1.44172, the edges of the second node have weights 0.707424, 1.3816 and 0.682372.

Note, that graph files allow the estimation of very general correlated effects based on 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 three-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\zambiasort.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
```

3.6 Simultaneous selection of variables and smoothing parameters

To perform simultaneous selection of variables and smoothing parameters in regression models, we first create a *stepwisereg object*:

```
> stepwisereg s
```

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 *stepwisereg 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 *stepwisereg objects*:

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

Note, that `outfile` does not only specify a directory but also a base filename (the character 's' 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 's'. Of course the base filename may be different from the name of the *stepwisereg object*.

In addition to parameter estimates of the selected model, *BayesX* also gives some further information on the selection process. Not all of this information is stored automatically but is printed in the *output window*. Therefore it is useful to store the contents of the *output window*. This is achieved by opening a log file using the `logopen` command

```
> logopen, replace using c:\data\logstep.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, therefore, are modeled nonparametrically (using P-splines with second order difference penalty in our example). However, a linear effect could be more appropriate and, hence, the linear effect is also considered for model selection. 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 modeled by a (quadratic) pairwise difference penalty, where the neighborhood matrix and possible weights associated with the neighbors are obtained from the *map object* `m`. The uncorrelated part is modelled by an i.i.d. Gaussian random effect (simple ridge type penalty). For each variable and function, the selection procedure additionally considers the possibility of removing the term from the model. Both effect variables `edu1` and `edu2` represent the overall effect of the categorical variable `edu`. Therefore the variable `edu` is specified as a `factor` for the selection procedure.

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

```
> s.regress hazstd = rcw + edu(factor) + tpr + sex + bmi(psplinerw2)
  + agc(psplinerw2) + district(spatial,map=m) + district(random),
  family=gaussian predict using d
```

If option `predict` is specified estimates for the linear predictor and the expectation of every observation are obtained.

```
  hazstd = const + tpr + sex + edu_1 + edu_2 + bmi +
           agc(psplinerw2,df=10.959,(lambda=15.4071)) +
           district(spatial,df=24.3664,(lambda=7.5775)) +
           district(random,df=17.8721,(lambda=35.6851))
  AIC_imp = -1024.7455

  Startmodel:

  hazstd = const + tpr + sex + edu_1 + edu_2 + bmi +
           agc(psplinerw2,df=10.959,(lambda=15.4071)) +
           district(spatial,df=24.3664,(lambda=7.5775)) +
           district(random,df=17.8721,(lambda=35.6851))
  AIC_imp = -1024.4848

  Startmodel:

  hazstd = const + tpr + sex + edu_1 + edu_2 + bmi +
           agc(psplinerw2,df=10.959,(lambda=15.4071)) +
           district(spatial,df=24.3664,(lambda=7.5775)) +
           district(random,df=17.8721,(lambda=35.6851))
  AIC_imp = -1024.4091

  Final Model:

  hazstd = const + tpr + sex + edu_1 + edu_2 + bmi +
           agc(psplinerw2,df=10.959,(lambda=15.4071)) +
           district(spatial,df=24.3664,(lambda=7.5775)) +
           district(random,df=17.8721,(lambda=35.6851))
  AIC_imp = -1024.3814

  Used number of iterations: 10

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

  Final Model:

  hazstd = const + tpr + sex + edu_1 + edu_2 + bmi +
           agc(psplinerw2,df=10.959,(lambda=15.4071)) +
           district(spatial,df=24.3664,(lambda=7.5775)) +
           district(random,df=17.8721,(lambda=35.6851))
  AIC_imp = -1024.352


RESPONSE DISTRIBUTION:

  Gaussian
  Number of observations: 4847


ESTIMATION RESULTS:


  Predicted values:

  Estimated mean of predictors, expectation of response and
  individual deviances are stored in file
  c:\bayesx\output\s_predictmean.raw


  Saturated deviance: 4847
```

```
Estimation results for the scale parameter:

sigma2:         0.789725



FixedEffects1

Variable  mean           Std. Dev.      2.5% quant.    median         97.5% quant.
const     -0.422865      0              0              0              0
tpr       0.0945051      0              0              0              0
sex       -0.0589084     0              0              0              0
edu_1     -0.063032      0              0              0              0
edu_2     0.234741       0              0              0              0
bmi       0.0209163      0              0              0              0

Results for fixed effects are also stored in file
c:\bayesx\output\s_FixedEffects1.res


f_agc

Results are stored in file
c:\bayesx\output\s_f_agc_pspline.res

Results may be visualized using the R / S-Plus function 'plotnonp'
Type for example:
plotnonp("c:\\bayesx\\output\\s_f_agc_pspline.res")


f_district

Results are stored in file
c:\bayesx\output\s_f_district_spatial.res

Results may be visualized using the R / S-Plus function
'drawmap'


f_district

Results for random effects are stored in file
c:\bayesx\output\s_f_district_random.res

Results for the sum of the structured and unstructured
spatial effects are stored in file
c:\bayesx\output\s_district_spatialtotal.res

Files of model summary:

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

Batch file for visualizing effects of nonlinear functions is stored in file
c:\bayesx\output\s_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:\bayesx\output\s_splus.txt
```

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

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

Latex file of model summary is stored in file
c:\bayesx\output\s_model_summary.tex

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

Note that variable *rcw* was removed from the model and variable *bmi* is modelled by a linear effect.

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. The files contain the posterior mean. For example, the beginning of file `c:\data\s_f_age_pspline.res` for the effect of `age` looks like this:

intnr	age	pmean	pqu2p5	pqu10	pmed	pqu90	pqu97p5	pcat95	pcat80
1	0	1.0981	0	0	0	0	0	0	0
2	1	0.978917	0	0	0	0	0	0	0
3	2	0.867664	0	0	0	0	0	0	0

Note, that columns *pmean* to *pcat80* contain no values. Methods for obtaining values for median, posterior 2.5%, 10%, 90% and 97.5% quantiles, and corresponding 95% and 80% posterior probabilities of the estimated effects are described in section 3.7.

Some nonparametric effects are visualized by *BayesX* automatically and the resulting graphs are stored in ps format. E.g. the effect of `age` is visualized in the file `c:\data\s_f_age_pspline.ps` (compare the results on the previous pages for the other filenames). In addition to the postscript files a file containing the commands to reproduce the graphics is stored in the output directory. In our example the name of the file is `c:\data\s_graphics.prg`. The advantage is that additional options may be added by the user to customize the graphs (compare the respective two subsubsections).

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.

There are two results files containing information about the selection process. File `c:\data\s_models.raw` contains the base models used at the beginning of each iteration and the finally selected model. File `c:\data\s_criterium.raw` shows changes in the value of the selection criterion during the selection process.

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*.

3.7 Selection including calculation of confidence bands

stepwisereg objects allow both the calculation of conditional and unconditional confidence bands for nonlinear functions.

3.7.1 Conditional confidence bands

Conditional confidence bands are calculated conditional on the selected model, i.e. they are computed for selected variables and functions only. The computation of conditional confidence bands is based on an MCMC-algorithm subsequent to the selection procedure. For the selection of a

model with an adjacent computation of conditional confidence bands we can use method `regress` of *stepwisereg objects*:

```
> s.regress hazstd = rcw + edu(factor) + tpr + sex + bmi(psplinerw2)
  + agc(psplinerw2) + district(spatial,map=m) + district(random),
  CI=MCMCselect step=10 iterations=10000
  family=gaussian predict using d
```

Options `iterations` and `step` define properties of the MCMC-algorithm. The total number of MCMC iterations is given by `iterations`. Since, in general, these random numbers are correlated, we do not use all of them but thin out the Markov chain by the thinning parameter `step`. Specifying `step=10` as above forces *BayesX* to store only every 10th sampled parameter which leads to a random sample of length 1000 for every parameter in our example.

Note, that the choice of `iterations` also affects computation time. On a 2.4 GHz PC estimation of our model took about ? minute and ? seconds, which is rather fast in regard of the complexity of the model.

Now, file `c:\data\s_f_age_pspline.res` for the effect of `age` contains also results for median, posterior 2.5%, 10%, 90% and 97.5% quantiles, and corresponding 95% and 80% posterior probabilities of the estimated effects. The first few lines of the file looks like this:

intnr	age	pmean	pqu2p5	pqu10	pmed	pqu90	pqu97p5	pcat95	pcat80
1	0	1.09481	0.925783	0.984814	1.09623	1.20131	1.26294	1	1
2	1	0.977488	0.866704	0.906176	0.977659	1.04638	1.08583	1	1
3	2	0.867545	0.787211	0.815846	0.8677	0.918947	0.945463	1	1

The posterior quantiles and posterior probabilities may be changed by the user using the options `level1` and `level2`. For example specifying `level1=99` and `level2=70` in the options list of the `regress` command leads to the computation of 0.5%, 15%, 85% and 99.5% quantiles of the posterior. The defaults are `level1=95` and `level2=80`.

3.7.2 Unconditional confidence bands

Unconditional confidence bands consider the uncertainty due to model selection. Unconditional confidence bands are obtained using the following modified `regress` command:

```
> s.regress hazstd = rcw + edu(factor) + tpr + sex + bmi(psplinerw2)
  + agc(psplinerw2) + district(spatial,map=m) + district(random),
  CI=MCMCbootstrap bootstrapsamples=99 step=10 iterations=10000
  family=gaussian predict using d
```

Options `iterations` and `step` define properties of the MCMC-algorithm. The number of `iterations` is divided equally between the bootstrap iterations. Here, for each of the 99 bootstrap data sets (specified by *bootstrapsamples=99*) and the original data set 100 MCMC samples are drawn. The thinning parameter is `step =10`, i.e. every 10th sampled parameter per data set (i.e. 1000 samples altogether) are used for the computation of empirical quantiles. Note, that the number of bootstrap data sets strongly affects computation time.

In addition to confidence bands, a kind of sensitivity analysis regarding model selection can be performed using the estimation results. For the effect of `age`, file `c:\data\s_f_age_pspline_df.raw` contains the number of times each modelling alternative was selected during the bootstrapping process. This frequency distribution indicates the plausibility of the different modelling alternatives, especially of the modelling alternative selected for the original data. The first few lines of file `c:\data\s_f_age_pspline_df.raw` looks like this:

df_value	sp_value	frequency	selected

```
3.95883   1767.34   1   -
5.0261    634.521   21  -
5.96806   297.13    12  -
```

3.8 Visualizing estimation results

BayesX provides three alternatives for visualizing estimation results:

- As mentioned in the previous subsubsection, certain results are automatically visualized by *BayesX* and stored in postscript files.
- Post estimation commands of *stepwisereg objects* allow to visualize 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 visualization 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 subsubsection 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*. 3.9 describes how to customize plots.

3.8.1 Post estimation commands

After having estimated a regression model with subsequent calculation of unconditional confidence bands, i.e. having specified option *CI=MCMCbootstrap*, plots for nonparametric effects of continuous covariates can be produced using the post estimation command `plotnonp`:

```
> s.plotnonp 1
```

and

```
> s.plotnonp 2
```

produce the graphs shown in 3.4 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 subsubsection.

By default the plots contain the posterior mean and pointwise credible intervals according to the levels specified in the `regress` command. So by default the plot includes pointwise 80% and 95% credible intervals. Note, however, that for a simple model selection without calculation of credible bands only the effect itself can be plotted.

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

```
> s.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 visualized by drawing the respective map and coloring the regions of the map according to some characteristic of the posterior, e.g. the posterior mean. For the structured spatial effect this can be achieved using the post estimation command `drawmap`

```
> s.drawmap 3
```

which results in the graph shown in 3.5.

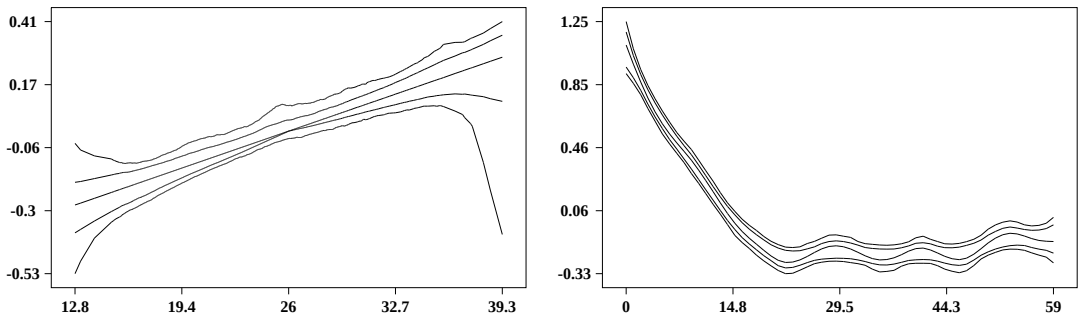


Figure 3.4: Effect of the body mass index of the child's mother and of the age of the child together with pointwise 80% and 95% credible intervals.

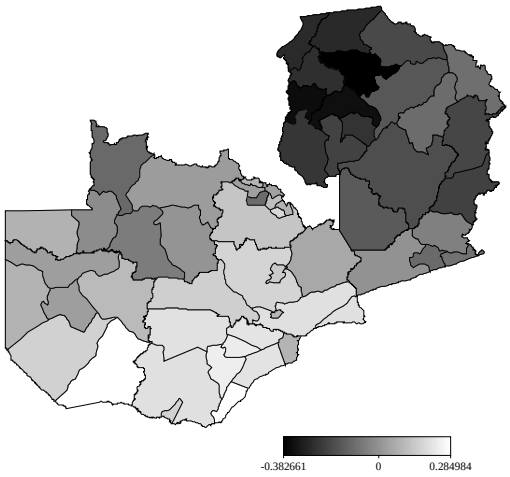


Figure 3.5: Posterior mean of the structured spatial effect.

3.8.2 Graph Objects

The commands presented in the previous paragraph work only after having estimated a regression model in the current *BayesX* session but it may also be useful to visualize results of former analyses. 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 paragraph 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\s_f_bmi_pspline.res
```

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

```
> graph g
```

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

```
> g.plot bmi pmean pqu2p5 pqu10 pqu90 pqu97p5 using res
```

reproduces the graph in 3.4.

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 visualize 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\s_f_district_spatial.res
```

stores the estimation results for the structured spatial effect in the *dataset object* `res`. Now we can visualize the posterior mean using method `drawmap` of *graph objects* leading again to the graph shown in 3.5:

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

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

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

```
> g.drawmap pcat95 district, map=m using res
```

The result is shown in 3.6.

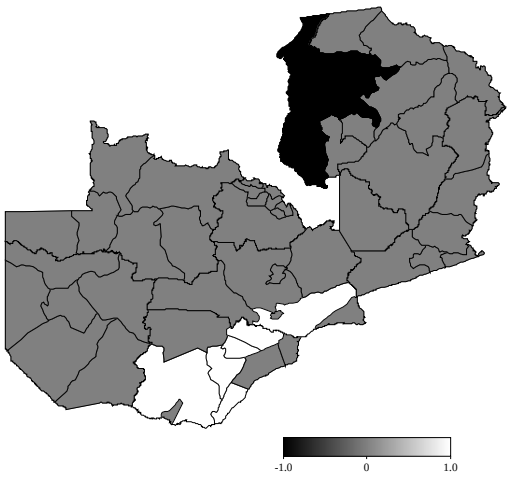


Figure 3.6: Posterior 95% probability of the structured spatial effect.

A further advantage of *graph objects* is, that they allow to visualize the estimation results for the uncorrelated spatial effects. Since these are modelled as unstructured random effects, *BayesX* is unable to recognize them as spatial effects. However, proceeding as follows gives us the possibility to plot the unstructured spatial effect shown in 3.7:

```
> res.infile using c:\data\s_f_district_random.res
> g.drawmap pmean district, map=m using res
```

3.9 Customizing graphics

This subsubsection describes how to customize 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 subsubsection 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

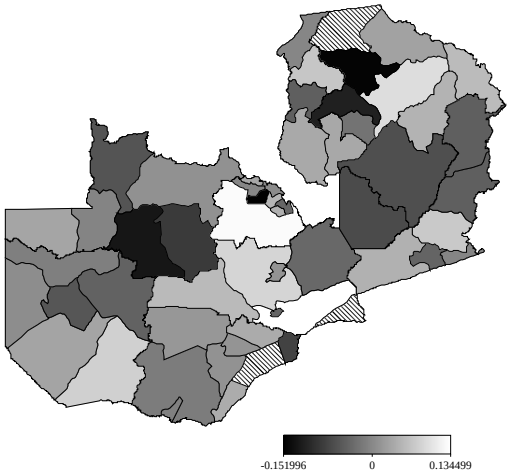


Figure 3.7: Posterior mean of the unstructured spatial effect.

intervals 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 3.7). If the default values of `level1` and `level2` have been used, specifying `levels=2` in the `plotnonp` command causes *BayesX* to plot the 80% credible interval only (3.8):

```
> s.plotnonp 2, levels=2
```

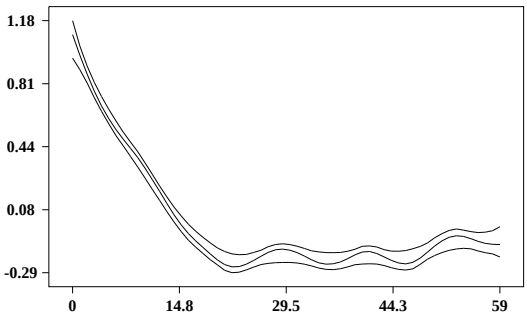


Figure 3.8: Effect of the age of the child with pointwise 80% credible interval only.

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 3.9 for the resulting plots):

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

By default *BayesX* displays x- and y-axis with five equidistant ticks according to the range of the data that is to be visualized. 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 3.10.

```
> s.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
```

3.10 also includes a graph for the effect of the age of the child that is customized in the same way

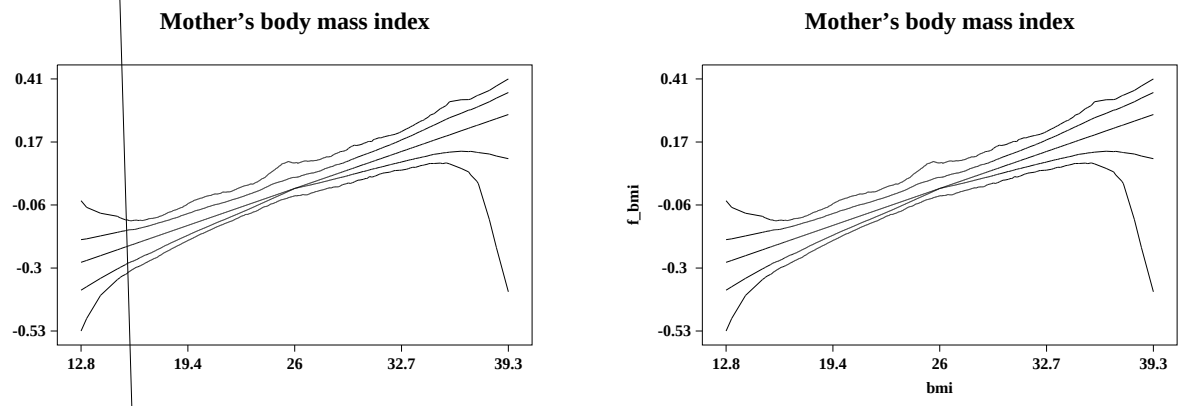


Figure 3.9: Specification of title and axis labels.

as for the effect of `bmi`.

```
> s.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
```

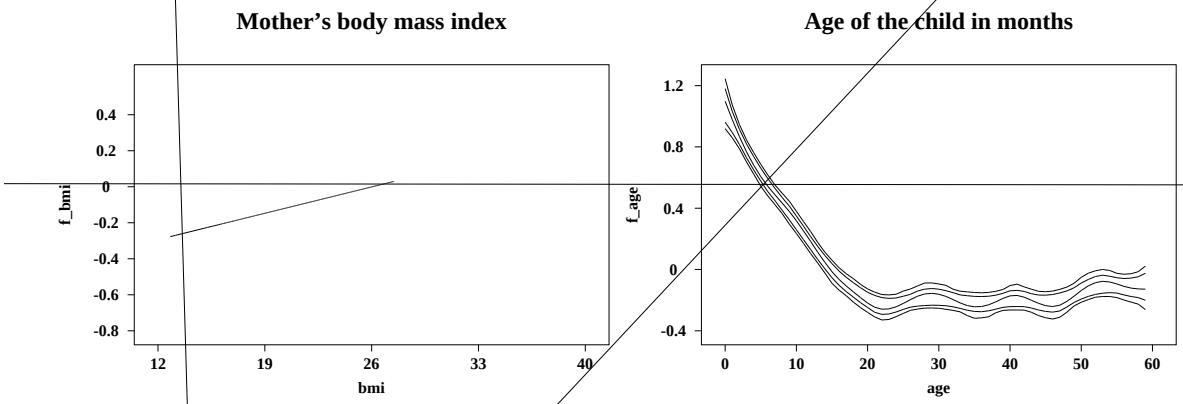


Figure 3.10: Re-defining x- and y-axis.

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

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

leads to the graph shown in 3.11 with higher values being represented through red colors and smaller values through green colors.

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

```
> s.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`:

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

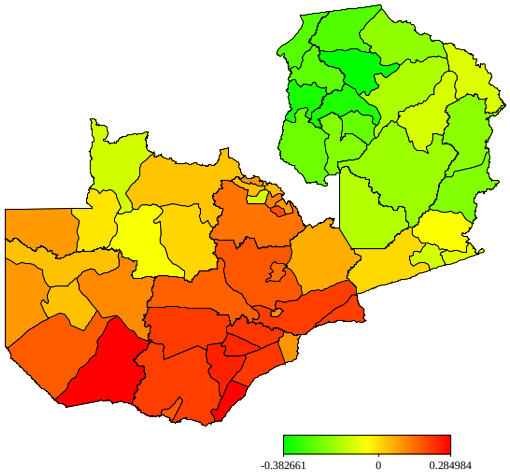


Figure 3.11: Posterior mean of the structured spatial effect in color.

The graph produced by the second command is shown in 3.12.

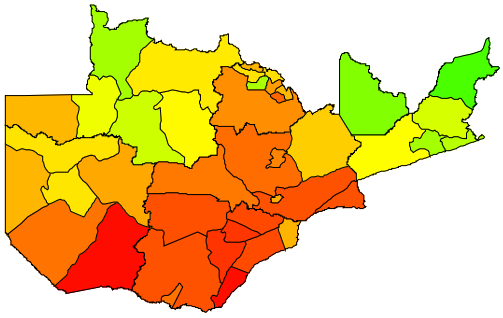


Figure 3.12: Specifying a title and the range of the plot for spatial effects.

Bibliography

Belitz, C. & Lang, S. (2008): Simultaneous selection of variables and smoothing parameters in structured additive regression models. *Computational Statistics and Data Analysis*, **53**, 61–81.

Brezger, A. & Lang, S., 2006: Generalized structured additive regression based on Bayesian P-splines. *Computational Statistics and Data Analysis*, **50**, 967–991.

Fahrmeir, L., Kneib, T. & Lang, S., 2004: Penalized structured additive regression for space-time data: A Bayesian perspective, *Statistica Sinica*, **14**, 731–761 .

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

Fahrmeir, L. & Osuna, L. (2006), Structured additive regression for overdispersed and zero-inflated count data. *Applied Stochastic Models in Business and Industry*, **22**, 351–369.

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

Hennerfeind, A., Brezger, A. & Fahrmeir, L., 2003: Geoadditive survival models. *Journal of the American Statistical Association*, **101**, 1065–1075

Kandala, N. B., Lang, S., Klasen, S. & 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.

Kneib, T. (2006) Geoadditive hazard regression for interval censored survival times. *Computational Statistics and Data Analysis*, **51**, 777–792.

Kneib, T. & Fahrmeir, L. (2006) Structured additive regression for categorical space-time data: A mixed model approach. *Biometrics*, **62**, 109–118.

Kneib, T. & Fahrmeir, L. (2007) A mixed model approach for geoadditive hazard regression. *Scandinavian Journal of Statistics*, **34** 207-228.

Lang, S. & Brezger, A., 2004: Bayesian P-splines. *Journal of Computational and Graphical Statistics*, **13**, 183–212.

Spiegelhalter, D.J., Best, N.G., Carlin, B.P. & van der Linde, A. (2002): Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society B*, **65**, 583–639