Advanced Bayesian Methods: Theory and Applications in R

Generic Basis Function Framework

Nikolaus Umlauf

https://nikum.org/abm.html

Addtive Models

- Consider observations $(y_i, x_{i1}, \ldots, x_{ik})$, $i = 1, \ldots, n$, of a continuous response variable y and covariates $x_1, \ldots, x_k$, whose effect on y can be modeled through a linear predictor.
- Additionally, we have observations $(z_{i1}, \ldots, z_{iq})$, $i = 1, \ldots, n$, of continuous covariates $z_1, \ldots, z_q$ whose effects are to be modeled and analyzed nonparametrically.

Addtive Models

- Additive models are defined by

$$\begin{aligned} y_i &= f_1(z_{i1}) + \ldots + f_q(z_{iq}) + \beta_0 + \beta_1 x_{i1} + \ldots + \beta_k x_{ik} + \varepsilon_i \\ &= f_1(z_{i1}) + \ldots + f_q(z_{iq}) + \eta_i^{lin} + \varepsilon_i \\ &= \eta_i^{add} + \varepsilon_i \end{aligned}$$

with

$$\eta_i^{lin} = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_k x_{ik}, \qquad \eta_i^{add} = f_1(z_{i1}) + \ldots + f_q(z_{iq}) + \eta_i^{lin}.$$

Addtive Models

- There are two points to note about additive models. Firstly, the assumption of additive effects is a fairly strong one, e.g., $f_1(z_1) + f_2(z_2)$ is a quite restrictive case of the general smooth function of two variables $f(z_1, z_2)$.
- Secondly, the fact that the model contains more than one function introduces an identifiability problem.
- If we add a constant $c \neq 0$ to the function $f_1(z_1)$ and subtract at the same time c from a second function $f_2(z_2)$, the sum

$$f_1(z_1) + f_2(z_2) = f_1(z_1) + c + f_2(z_2) - c$$

  remains the same, i.e. the predictor η does not change if $f_1(z_1)$ changes to $\tilde{f}_1(z_1) = f_1(z_1) + c$ and $f_2(z_2)$ changes to $\tilde{f}_2(z_2) = f_2(z_2) - c$.

Addtive Models

- Although the functional form of $f_1(z_1)$ and $\tilde{f}_1(z_1)$ or $f_2(z_2)$ and $\tilde{f}_2(z_2)$ is unchanged, the overall level of the functions is arbitrary without imposing some further restrictions.
- Hence it is necessary to fix the level of the functions. This is usually obtained by “centering the functions around zero", such that

$$\sum_{i=1}^{n} f_1(z_{i1}) = \ldots = \sum_{i=1}^{n} f_q(z_{iq}) = 0.$$

- Provided that the identifiability issue is addressed, the additive model can be represented using penalized splines, estimated by penalized least squares and the degree of the smoothing parameter selected by cross validation.

Addtive Models

- As with the linear model, the assumptions regarding the error variables ε_i carry over to the response variables y_i, i.e. the y_i are (conditionally) independent with

$$\begin{aligned} E(y_i) &= \mu_i = f_1(z_{i1}) + \ldots + f_q(z_{iq}) + \beta_0 + \beta_1 x_{i1} + \ldots + \beta_k x_{ik} \\ Var(y_i) &= \sigma^2 \end{aligned}$$

  and, if applicable, normally distributed with

$$y_i \sim N(\mu_i, \sigma^2).$$

- The literature often refers the special case

$$y_i = f_1(z_{i1}) + \ldots + f_q(z_{iq}) + \varepsilon_i,$$

  without the additional linear predictor η_i^{lin} as the additive model.

Generic Framework

- Penalized splines are one representative for a variety of effects that can be cast into a generic basis function framework.
- Let $\boldsymbol{x}_i$ denote some generic type of covariate information and assume

$$f(\boldsymbol{x}_i) = \sum_{l=1}^{L} \gamma_l B_l(\boldsymbol{x}_i)$$

  with L basis functions $B_l(\boldsymbol{x}_i)$.
- The vector of function evaluations $\boldsymbol{f} = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n))^\top$ at the observed covariate values is then given by

$$\boldsymbol{f} = \boldsymbol{B}\boldsymbol{\gamma},$$

  where $\boldsymbol{B}$ is the design matrix obtained from the basis function evaluations and γ is the corresponding vector of basis coefficients.

Generic Framework

- To regularize estimation, assign the prior

$$p(\boldsymbol{\gamma}|\tau^2) \propto \left(\frac{1}{\tau^2}\right)^{\text{rk}(\boldsymbol{K})/2} \exp\left(-\frac{1}{2\tau^2}\boldsymbol{\gamma}'\boldsymbol{K}\boldsymbol{\gamma}\right)$$

  with positive semidefinite prior precision matrix $\boldsymbol{K}$ and prior variance parameter τ^2.
- An effect is then characterized by the chosen basis functions $B_l(\boldsymbol{x})$, the structure of the precision matrix $\boldsymbol{K}$ and the hyperprior assumed for τ^2
- For the latter, it is common to use conjugate inverse gamma hyperpriors.

Generic Framework

- The posterior of an additive model is then given by

$$
\begin{aligned}
p(\boldsymbol{\theta}|\mathbf{y}) \quad \propto \quad & \frac{1}{\left(\sigma^2\right)^{\frac{n}{2}}} \exp\left(-\frac{1}{2\sigma^2}\left(\mathbf{y}-\boldsymbol{\eta}\right)^\top\left(\mathbf{y}-\boldsymbol{\eta}\right)\right) \\
& \prod_{j=1}^{q} \frac{1}{\left(\tau_j^2\right)^{\text{rk}\mathbf{K}_j)/2}} \exp\left(-\frac{1}{2\tau_j^2}\boldsymbol{\gamma}_j^\top \mathbf{K}_j \boldsymbol{\gamma}_j\right) \\
& \prod_{j=1}^{q} \left(\tau_j^2\right)^{-a_j-1} \exp\left(-\frac{b_j}{\tau_j^2}\right) \left(\sigma^2\right)^{-a_{\sigma^2}-1} \exp\left(-\frac{b_{\sigma^2}}{\sigma^2}\right).
\end{aligned}
$$

Generic Framework

- Similar to the normal linear regression model we derive

$$\gamma_j|\cdot \sim N(\mu_j, \mathbf{\Sigma}_j)$$

with expectation and covariance matrix

$$\begin{aligned}
\mu_j &= E(\gamma_j|\cdot) = \left(\frac{1}{\sigma^2}\mathbf{Z}_j^\top\mathbf{Z}_j + \frac{1}{\tau_j^2}\mathbf{K}_j\right)^{-1}\frac{1}{\sigma^2}\mathbf{Z}_j^\top(\mathbf{y} - \eta_{-j}) \\
\mathbf{\Sigma}_j &= Cov(\gamma_j|\cdot) = \left(\frac{1}{\sigma^2}\mathbf{Z}_j^\top\mathbf{Z}_j + \frac{1}{\tau_j^2}\mathbf{K}_j\right)^{-1}.
\end{aligned}$$

Generic Framework

- The full conditional of the parametric coefficients has expectation and covariance matrix

$$\begin{aligned}\boldsymbol{\mu}_\beta &= E(\boldsymbol{\beta}|\cdot) = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top(\mathbf{y} - (\boldsymbol{\eta} - \mathbf{Z}_1\boldsymbol{\gamma}_1 - \ldots - \mathbf{Z}_q\boldsymbol{\gamma}_q)) \\ \boldsymbol{\Sigma}_\beta &= Cov(\boldsymbol{\beta}|\cdot) = \frac{1}{\sigma^2}(\mathbf{X}^\top\mathbf{X})^{-1}.\end{aligned}$$

- For the variance parameters, we obtain

$$\begin{aligned}\tau_j^2|\cdot &\sim IG(a_j + 0.5\text{rk}(\mathbf{K}_j)), b_j + 0.5\boldsymbol{\gamma}_j^\top\mathbf{K}_j\boldsymbol{\gamma}_j), \\ \sigma^2|\cdot &\sim IG(a_{\sigma^2} + 0.5n, b_{\sigma^2} + 0.5(\mathbf{y} - \boldsymbol{\eta})^\top(\mathbf{y} - \boldsymbol{\eta})).\end{aligned}$$

Efficient Sampling from Multivariate Normals

- Naively working with multivariate normal distributions can lead to large computation times.
- Particular bottleneck: The covariance matrix $\boldsymbol{\Sigma}_j$ (and its inverse).
- Efficient simulation from $N(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j^{-1})$ avoiding the explicit computation of $\boldsymbol{\Sigma}_j$:
  1. Compute the Cholesky factorisation $\boldsymbol{\Sigma}_j^{-1} = \boldsymbol{L}\boldsymbol{L}^\top$.
  2. Sample $\boldsymbol{z} \sim \mathrm{N}(\boldsymbol{0}, \boldsymbol{I})$.
  3. Solve $\boldsymbol{L}^\top \boldsymbol{v} = \boldsymbol{z}$.
  4. Return $\boldsymbol{\mu}_j + \boldsymbol{v}$.
- Can be extended to also determine $\boldsymbol{\mu}_j$ without computing $\boldsymbol{\Sigma}_j^{-1}$.
- In many cases, $\boldsymbol{\Sigma}_j^{-1}$ has a sparse matrix structure that can be exploited for further efficiency gains.

Smooth Terms in `mgcv`

- In the *mgcv* package, smooth terms are used to model complex nonlinear relationships between predictors and responses.
- Behind the scene, smooth terms are constructed using the `smoothCon()` function.
- The `smoothCon()` function accepts smooth specification objects such as `s()`, `te()`, and/or `ti()`, and utilizes them to generate the associated design and penalty matrices.
- Smooth terms can be used in GAM, GAMM and GAMLSS.

Smooth Terms in `mgcv`

Examples:

```
R> library("mgcv")
R> d <- data.frame("x" = seq(0, 1, length = 300))
```

Create smooth term.

```
R> sc <- smoothCon(s(x, bs = "ps", k = 10), d, knots = NULL)[[1]]
R> print(names(sc))
 [1] "term"           "bs.dim"         "fixed"          "dim"
 [5] "p.order"        "by"             "label"          "xt"
 [9] "id"             "sp"             "deriv"          "X"
[13] "mono"           "D"              "S"              "rank"
[17] "null.space.dim" "knots"          "m"              "plot.me"
[21] "side.constrain" "repara"         "C"              "df"
[25] "S.scale"
```

Smooth Terms in mgcv

Design matrix.

```
R> print(head(sc$X))
          [,1]      [,2]      [,3]         [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,] 0.1631980 0.6666180 0.1701839 5.682503e-08    0    0    0    0    0     0
[2,] 0.1519473 0.6657595 0.1822886 4.659653e-06    0    0    0    0    0     0
[3,] 0.1412258 0.6638588 0.1948895 2.583111e-05    0    0    0    0    0     0
[4,] 0.1310210 0.6609543 0.2079483 7.632610e-05    0    0    0    0    0     0
[5,] 0.1213200 0.6570842 0.2214268 1.688995e-04    0    0    0    0    0     0
[6,] 0.1121101 0.6522869 0.2352867 3.163063e-04    0    0    0    0    0     0
```

Smooth Terms in `mgcv`

Penalty matrix.

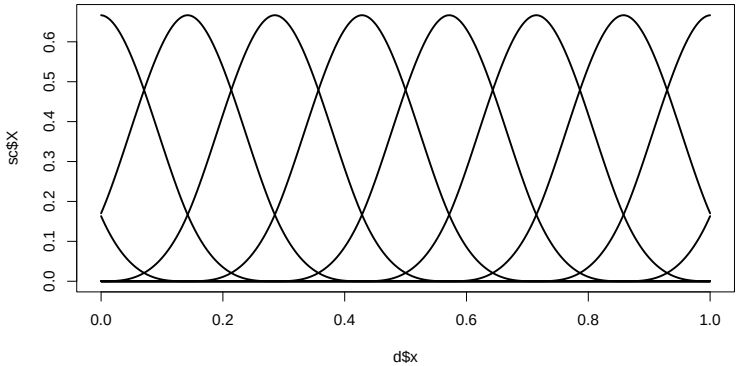
```
R> print(head(sc$S[[1]]))
```

```
        [,1]    [,2]    [,3]    [,4]    [,5]    [,6]    [,7]    [,8] [,9] [,10]
[1,]  0.0625 -0.1250  0.0625  0.0000  0.0000  0.0000  0.0000  0.0000    0     0
[2,] -0.1250  0.3125 -0.2500  0.0625  0.0000  0.0000  0.0000  0.0000    0     0
[3,]  0.0625 -0.2500  0.3750 -0.2500  0.0625  0.0000  0.0000  0.0000    0     0
[4,]  0.0000  0.0625 -0.2500  0.3750 -0.2500  0.0625  0.0000  0.0000    0     0
[5,]  0.0000  0.0000  0.0625 -0.2500  0.3750 -0.2500  0.0625  0.0000    0     0
[6,]  0.0000  0.0000  0.0000  0.0625 -0.2500  0.3750 -0.2500  0.0625    0     0
```

Smooth Terms in mgcv

Plot the design matrix.

```
R> par(mar = c(4, 4, 0.5, 0.5))
R> matplot(d$x, sc$X, type = "l", lty = 1, col = 1, lwd = 2)
```



Smooth Terms in `mgcv`

- Formula syntax:

Linear effects $\mathbf{X}\boldsymbol{\beta}$	`x1 + x2 + x3`
Nonlinear effects $f(\mathbf{x}) = f(x_1)$	`s(x1)`
Interaction surfaces $f(\mathbf{x}) = f(x_1, x_2)$	`s(x1,x2)`, `te(x1,x2)` or `ti(x1,x2)`
Discrete Spatial $f(\mathbf{x}) = f_{\text{spat}}(x_s)$	`s(xs,bs="mrf",xt=list(penalty=K))`
Varying coefficients $f(\mathbf{x}) = x_1 s(x_2)$	`s(x2,by=x1)`
Spatially varying effects $f(\mathbf{x}) = x_1 f_{\text{spat}}(x_s)$,	`s(xs,bs="mrf",xt=list(penalty=K),by=x1),`
or $f(\mathbf{x}) = x_1 f(x_2, x_3)$	`s(x2,x3,by=x1)` or `te(x2,x3,by=x1)`
Random intercepts with clusters c: $f(\mathbf{x}) = \beta_c$	`s(id,bs="re")`
Random slopes with clusters c: $f(\mathbf{x}) = x_1 \beta_c$	`s(id,x1,bs="re")`

Spatial *bamlss*

Example:

London fire data.

```
R> library("bamlss")
R> data("LondonFire")
R> head(LondonFire)
```

```
                 coordinates arrivaltime   daytime  fsintens
12279 (-0.09832153, 51.65413)    6.033333 0.1263889  248.6067
12280 (-0.04467665, 51.48959)    3.400000 0.3266667 1646.1975
12281  (-0.1101969, 51.47268)    4.383333 0.4641667 1333.3776
12282  (-0.2448571, 51.45319)    5.800000 1.9297222  300.6900
12283  (-0.1874054, 51.48648)    5.133333 1.9308333 1195.7156
12284  (-0.2893525, 51.61031)    4.966667 3.5480556  319.6787
```

Spatial *bamlss*

Transform to data frame.

```
R> library("sp")
R> d <- as.data.frame(LondonFire)
R> head(d)
```

```
      arrivaltime   daytime  fsintens         lon      lat
12279    6.033333 0.1263889  248.6067 -0.09832153 51.65413
12280    3.400000 0.3266667 1646.1975 -0.04467665 51.48959
12281    4.383333 0.4641667 1333.3776 -0.11019694 51.47268
12282    5.800000 1.9297222  300.6900 -0.24485711 51.45319
12283    5.133333 1.9308333 1195.7156 -0.18740538 51.48648
12284    4.966667 3.5480556  319.6787 -0.28935255 51.61031
```

Spatial *bamlss*

Estimate spatial GAM.

```
R> f <-  arrivaltime ~ s(daytime, bs = "cc") + s(fsintens) + s(lon, lat, k = 30)

R> set.seed(123)
R> b <- bamlss(f, data = d)
AICc 24041.09 logPost -12037.1 logLik -11986.4 edf 33.888 eps 0.1067 iteration   1
AICc 24026.33 logPost -12037.0 logLik -11975.0 edf 37.833 eps 0.0078 iteration   2
AICc 24021.06 logPost -12035.5 logLik -11972.0 edf 38.238 eps 0.0012 iteration   3
AICc 24017.16 logPost -12034.2 logLik -11969.8 edf 38.464 eps 0.0008 iteration   4
AICc 24013.91 logPost -12033.1 logLik -11968.0 edf 38.654 eps 0.0007 iteration   5
AICc 24011.09 logPost -12032.2 logLik -11966.4 edf 38.825 eps 0.0006 iteration   6
AICc 24008.61 logPost -12031.5 logLik -11965.0 edf 38.982 eps 0.0005 iteration   7
AICc 24006.39 logPost -12030.8 logLik -11963.7 edf 39.129 eps 0.0005 iteration   8
AICc 24004.40 logPost -12030.3 logLik -11962.6 edf 39.267 eps 0.0004 iteration   9
AICc 24002.61 logPost -12029.8 logLik -11961.6 edf 39.397 eps 0.0004 iteration  10
AICc 24000.97 logPost -12029.4 logLik -11960.6 edf 39.520 eps 0.0004 iteration  11
AICc 23999.48 logPost -12029.1 logLik -11959.8 edf 39.636 eps 0.0003 iteration  12
```

Spatial *bamlss*

```
AICc 23998.11 logPost -12028.8 logLik -11959.0 edf 39.745 eps 0.0003 iteration  13
AICc 23996.86 logPost -12028.5 logLik -11958.3 edf 39.849 eps 0.0003 iteration  14
AICc 23995.70 logPost -12028.3 logLik -11957.6 edf 39.947 eps 0.0003 iteration  15
AICc 23994.63 logPost -12028.1 logLik -11956.9 edf 40.040 eps 0.0002 iteration  16
AICc 23993.64 logPost -12027.9 logLik -11956.4 edf 40.128 eps 0.0002 iteration  17
AICc 23992.71 logPost -12027.8 logLik -11955.8 edf 40.211 eps 0.0002 iteration  18
AICc 23991.86 logPost -12027.6 logLik -11955.3 edf 40.290 eps 0.0002 iteration  19
AICc 23991.06 logPost -12027.5 logLik -11954.8 edf 40.364 eps 0.0002 iteration  20
AICc 23990.31 logPost -12027.4 logLik -11954.4 edf 40.435 eps 0.0002 iteration  21
AICc 23989.61 logPost -12027.3 logLik -11954.0 edf 40.502 eps 0.0002 iteration  22
AICc 23988.96 logPost -12027.3 logLik -11953.6 edf 40.566 eps 0.0001 iteration  23
AICc 23988.34 logPost -12027.2 logLik -11953.2 edf 40.626 eps 0.0001 iteration  24
AICc 23987.76 logPost -12027.2 logLik -11952.9 edf 40.684 eps 0.0001 iteration  25
AICc 23987.22 logPost -12027.1 logLik -11952.5 edf 40.738 eps 0.0001 iteration  26
AICc 23986.71 logPost -12027.1 logLik -11952.2 edf 40.790 eps 0.0001 iteration  27
AICc 23986.22 logPost -12027.0 logLik -11951.9 edf 40.840 eps 0.0001 iteration  28
AICc 23985.77 logPost -12027.0 logLik -11951.7 edf 40.887 eps 0.0001 iteration  29
AICc 23985.33 logPost -12027.0 logLik -11951.4 edf 40.932 eps 0.0001 iteration  30
```

Spatial *bamlss*

```
AICc 23984.93 logPost -12026.9 logLik -11951.1 edf 40.974 eps 0.0001 iteration  31
AICc 23984.54 logPost -12026.9 logLik -11950.9 edf 41.015 eps 0.0001 iteration  32
AICc 23984.17 logPost -12026.9 logLik -11950.7 edf 41.054 eps 0.0001 iteration  33
AICc 23984.17 logPost -12026.9 logLik -11950.7 edf 41.054 eps 0.0001 iteration  33
elapsed time:  2.46sec
Starting the sampler...

|                    |   0% 13.33sec
|*                   |   5% 12.71sec  0.67sec
|**                  |  10% 12.09sec  1.34sec
|***                 |  15% 11.44sec  2.02sec
|****                |  20% 10.92sec  2.73sec
|*****               |  25% 10.40sec  3.47sec
|******              |  30%  9.80sec  4.20sec
|*******             |  35%  9.24sec  4.98sec
|********            |  40%  8.73sec  5.82sec
|*********           |  45%  8.00sec  6.55sec
|**********          |  50%  7.27sec  7.27sec
```

Spatial *bamlss*

```
|***********          |  55%  6.55sec  8.00sec
|************         |  60%  5.82sec  8.74sec
|*************        |  65%  5.10sec  9.47sec
|**************       |  70%  4.39sec 10.23sec
|***************      |  75%  3.66sec 10.97sec
|****************     |  80%  2.93sec 11.73sec
|*****************    |  85%  2.21sec 12.54sec
|******************   |  90%  1.48sec 13.29sec
|******************** |  95%  0.74sec 14.01sec
|*********************| 100%  0.00sec 14.75sec
```

Spatial *bamlss*

Model summary.

```
R> summary(b)

Call:
bamlss(formula = f, data = d)
---
Family: gaussian
Link function: mu = identity, sigma = log
*---
Formula mu:
---
arrivaltime ~ s(daytime, bs = "cc") + s(fsintens) + s(lon, lat,
    k = 30)
-
Parametric coefficients:
             Mean  2.5%   50% 97.5% parameters
(Intercept) 5.328 5.281 5.328 5.378      5.329
-
```

Spatial *bamlss*

```
Acceptance probability:
      Mean 2.5% 50% 97.5%
alpha    1    1   1     1
-
Smooth terms:
                        Mean      2.5%       50%     97.5% parameters
s(daytime).tau21  1.153e-02 2.670e-03 8.388e-03 3.897e-02      0.016
s(daytime).alpha  1.000e+00 1.000e+00 1.000e+00 1.000e+00         NA
s(daytime).edf    5.851e+00 4.635e+00 5.840e+00 7.105e+00      6.448
s(fsintens).tau21 4.337e+01 9.229e+00 3.459e+01 1.290e+02     46.121
s(fsintens).alpha 1.000e+00 1.000e+00 1.000e+00 1.000e+00         NA
s(fsintens).edf   7.652e+00 6.357e+00 7.729e+00 8.545e+00      7.969
s(lon,lat).tau21  5.379e+00 2.013e+00 4.962e+00 1.155e+01      3.896
s(lon,lat).alpha  1.000e+00 1.000e+00 1.000e+00 1.000e+00         NA
s(lon,lat).edf    2.513e+01 2.211e+01 2.536e+01 2.720e+01     24.638
---
Formula sigma:
---
```

Spatial *bamlss*

```
sigma ~ 1
-
Parametric coefficients:
             Mean   2.5%    50%  97.5% parameters
(Intercept) 0.6317 0.6146 0.6316 0.6509      0.628
-
Acceptance probability:
        Mean   2.5%    50% 97.5%
alpha 0.9950 0.9569 1.0000     1
---
Sampler summary:
-
DIC = 23977.76 logLik = -11968.1 pd = 41.5607
runtime = 14.881
---
Optimizer summary:
-
```

Spatial *bamlss*

```
AICc = 23984.18 edf = 41.0546 logLik = -11950.74
logPost = -12026.96 nobs = 5838 runtime = 2.463
```

```
R> DIC(b)
```

```
      DIC       pd
 23977.76 41.56065
```

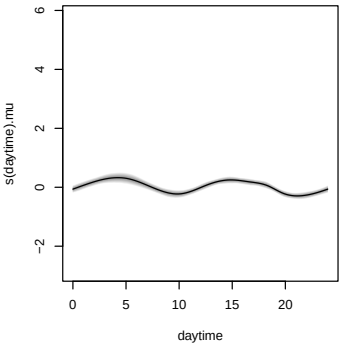
```
R> WAIC(b)
```

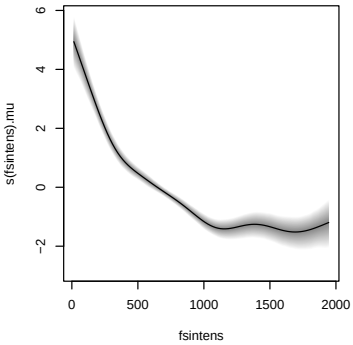
```
    WAIC1    WAIC2       p1       p2
 23985.32 23986.87 49.12139 49.89967
```

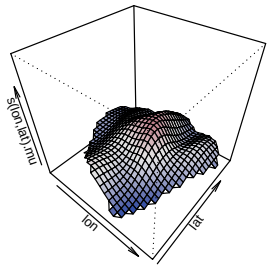
Spatial *bamlss*

Visualize effects.

```
R> par(mfrow = c(1, 3), mar = c(4, 4, 1, 1))
R> plot(b, pages = 0)
```







Spatial *bamlss*

MCMC samples.

```
R> par(mfrow = c(1, 2), mar = c(4, 4, 1, 1))
R> plot(b, which = "samples", model = "mu", term = "(Intercept)")
```