Universität Innsbruck

BAMLSS

Bayesian Additive Models for Location Scale and Shape (and Beyond)

Nikolaus Umlauf, Nadja Klein,
Stefan Lang, Achim Zeileis

http://eeecon.uibk.ac.at/~umlauf/

Overview

- Introduction
- Distributional regression
- General architecture
- R package **BayesR**
- Example

Introduction

A **not** complete list of software packages dealing with Bayesian regression models:

- **bayesm**, univariate and multivariate, SUR, multinomial logit, ...
- **bayesSurv**, survival regression, ...
- **MCMCpack**, linear regression, logit, ordinal probit, probit, Poisson regression, ...
- **MCMCglmm**, generalized linear mixed models (GLMM).
- **spikeSlabGAM**, Bayesian variable selection, model choice, in generalized additive mixed models (GAMM), ...
- **gammSlice**, generalized additive mixed models (GAMM).
- **BayesX**, structured additive distributional regression (STAR), ...
- **INLA**, generalized additive mixed models (GAMM), ...
- **WinBUGS**, **JAGS**, **STAN**, general purpose sampling engines.

$\vdots$

Introduction

Most Bayesian software packages provide support for the estimation of so called mixed models (random effects), i.e. incorporating linear predictors of the form

$$\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta} + \mathbf{U}\gamma,$$

where $\mathbf{X}\beta$ are fixed effects, e.g. $p(\boldsymbol{\beta}) \propto$ const, and $\mathbf{U}\gamma$ are the random effects, $\gamma \sim N(\mathbf{0}, \mathbf{Q}(\boldsymbol{\tau}^2))$.

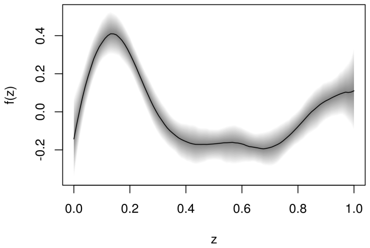
Few Bayesian software packages provide support for the estimation of semiparametric regression models with structured additive predictor

$$\eta = f_1(\mathbf{z}) + \ldots + f_p(\mathbf{z}) + \mathbf{x}^\top \boldsymbol{\beta},$$

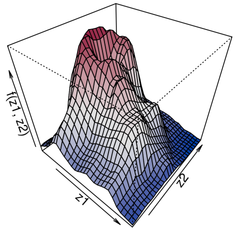
where f_j are possibly smooth functions and $\mathbf{z}$ represents a generic vector of all nonlinear modeled covariates.

Introduction

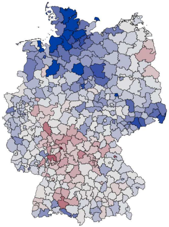
Nonlinear effects of continuous covariates



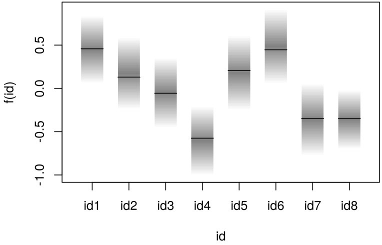
Two-dimensional surfaces



Spatially correlated effects f(z) = f(s)



Random intercepts

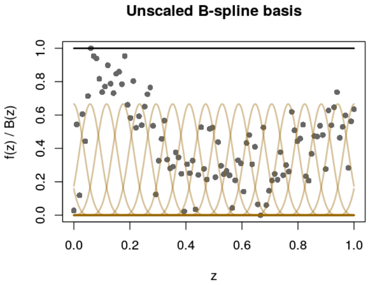


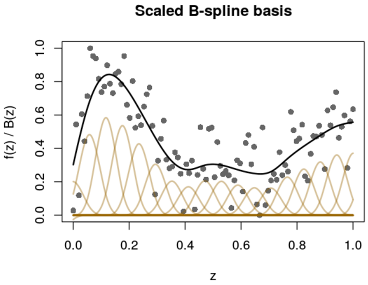
Introduction

Within the basis function approach, the vector of function evaluations $\mathbf{f}_j = (f_j(\mathbf{z}_1), \ldots, f_j(\mathbf{z}_n))$ of the $i = 1, \ldots, n$ observations can be written in matrix notation

$$\mathbf{f}_j = \mathbf{Z}_j \gamma_j,$$

with $\mathbf{Z}_j$ as the design matrix, where γ_j are unknown regression coefficients. Form of $\mathbf{Z}_j$ only depends on the functional type chosen.





Introduction

Penalized least squares:

$$\text{PLS}(\gamma, \boldsymbol{\lambda}) = ||\mathbf{y} - \boldsymbol{\eta}||^2 + \lambda_1 \gamma_1' \mathbf{K}_1 \gamma_1 + \ldots + \lambda_p \gamma_p' \mathbf{K}_p \gamma_p.$$

A general Prior for γ in the corresponding Bayesian approach

$$p(\gamma_j | \tau_j^2) \propto \exp\left(-\frac{1}{2\tau_j^2} \gamma_j' \mathbf{K}_j \gamma_j\right),$$

τ_j^2 variance parameter, governs the smoothness of f_j.

Structure of $\mathbf{K}_j$ also depends on the type of covariates and on assumptions about smoothness of $\mathbf{f}_j$.

The variance parameter τ_j^2 is equivalent to the inverse smoothing parameter in a frequentist approach.

Introduction

However, any basis function representation can be transformed into a mixed model representation

$$\mathbf{f}_j = \mathbf{Z}_j\boldsymbol{\gamma}_j = \mathbf{Z}_j(\tilde{\mathbf{X}}\boldsymbol{\beta} + \tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}}) = \mathbf{X}\boldsymbol{\beta} + \mathbf{U}\tilde{\boldsymbol{\gamma}},$$

with fixed effects $\boldsymbol{\beta}$ and random effects $\tilde{\boldsymbol{\gamma}} \sim N(\mathbf{0}, \tau^2\mathbf{I})$.

So the number of software packages that can estimate semiparametric models is actually quite large.

The number of different models that can be fit with these engines is even larger.

Introduction

The basic ideas are:

- Design a framework that makes it (a) easy to use different estimation engines and (b) fit models with a **structured additive predictor**.
- Therefore, we need to employ symbolic descriptions that do **not** restrict to any specific type of model and term structure.
- I.e., the aim is to use specialized/optimized engines to apply Bayesian **structured additive distributional regression** a.k.a. Bayesian additive models for location scale and shape (**BAMLSS**) and beyond.
- The approach should have **maximum flexibility/extendability**, also concerning functional types.

Distributional regression

Within this framework any parameter of a population distribution may be modeled by explanatory variables

$$\mathbf{y} \sim \mathcal{D}\left(g_1(\boldsymbol{\theta}_1) = \boldsymbol{\eta}_1,\ g_2(\boldsymbol{\theta}_2) = \boldsymbol{\eta}_2, \ldots,\ g_K(\boldsymbol{\theta}_K) = \boldsymbol{\eta}_K\right),$$

where $\mathcal{D}$ denotes any parametric distribution available for the response variable.

Each parameter is linked to a structured additive predictor

$$g_k(\boldsymbol{\theta}_k) = \boldsymbol{\eta}_k = \mathbf{Z}_{1k}\boldsymbol{\gamma}_{1k} + \ldots + \mathbf{Z}_{pk}\boldsymbol{\gamma}_{pk} + \mathbf{X}_k\boldsymbol{\beta}_k,\ \ k = 1, \ldots, K,$$

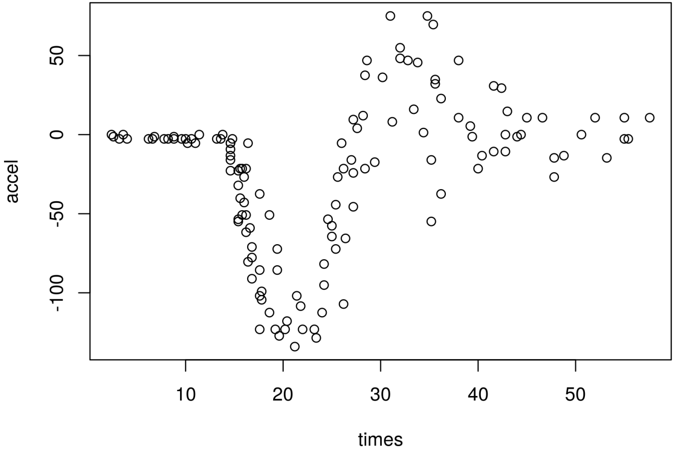
where $g_k(\cdot)$ are known monotonic link functions.

The observations y_i are assumed to be independent and conditional on a pre-specified parametric density $f(y_i|\boldsymbol{\theta}_{i1}, \ldots, \boldsymbol{\theta}_{iK})$.

Distributional regression

Example: Head acceleration in a simulated motorcycle accident

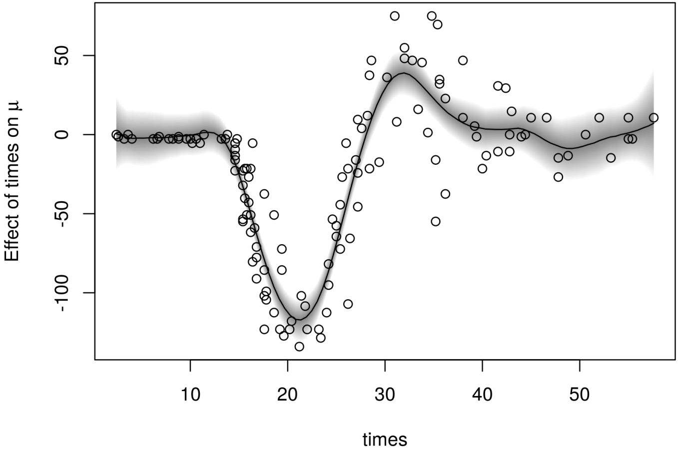
$$\texttt{accel} \sim N(\boldsymbol{\mu}, \boldsymbol{\sigma}^2).$$



Distributional regression

Example: Head acceleration in a simulated motorcycle accident

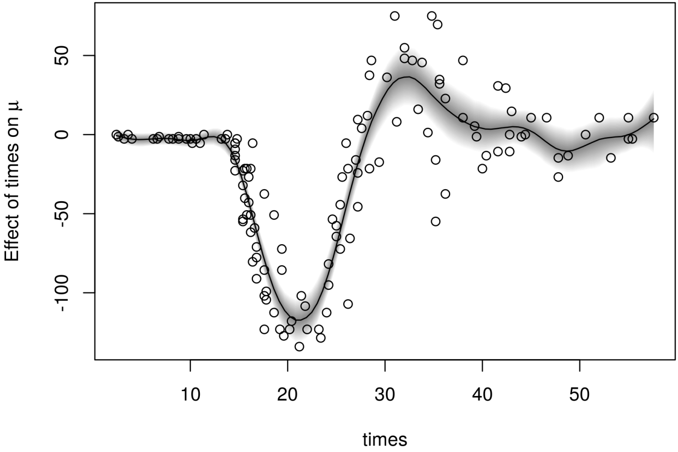
$$\texttt{accel} \sim N(\boldsymbol{\mu} = f(\texttt{times}), log(\boldsymbol{\sigma}^2) = \beta_0).$$



Distributional regression

Example: Head acceleration in a simulated motorcycle accident

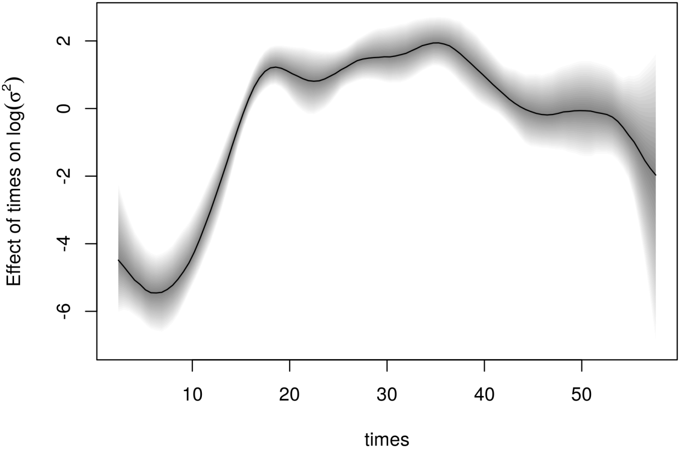
$$\texttt{accel} \sim N(\boldsymbol{\mu} = f(\texttt{times}), log(\boldsymbol{\sigma}^2) = f(\texttt{times})).$$



Distributional regression

Example: Head acceleration in a simulated motorcycle accident

$$\texttt{accel} \sim N(\boldsymbol{\mu} = f(\texttt{times}), log(\boldsymbol{\sigma}^2) = f(\texttt{times})).$$



Distributional regression

Sketch on MCMC inference

Based on the idea of generalized additive models for location, scale and shape (GAMLSS), which extends the exponential family regression framework to multi-parameter modeling.

Metropolis-Hastings based on iteratively weighted least squares proposals (IWLS):

$$\boldsymbol{\mu}_j = \mathbf{P}_j^{-1}\mathbf{Z}_j'\mathbf{W}(\mathbf{z} - \boldsymbol{\eta}_{-j}) \qquad \mathbf{P}_j = \mathbf{Z}_j'\mathbf{W}\mathbf{Z}_j + \frac{1}{\tau_j^2}\mathbf{K}_j$$

with working weights

$$\mathbf{W} = diag\left(E\left(-\frac{\partial^2 l}{\partial \eta_i^2}\right)\right)$$

and working observations

$$\mathbf{z} = \eta\mathbf{W}^{-1}\mathbf{v} \qquad \mathbf{v} = \frac{\partial l}{\partial \boldsymbol{\eta}}$$

Distributional regression

Sketch on MCMC inference

Set the number of iterations T and starting values for the parameters.

while(i < T) {

  for(k in 1:K) {

    for(j in 1:p) {

$$\boldsymbol{\gamma}^{\star} \sim N((\boldsymbol{\mu}_{\mathtt{j}}^{[\mathtt{k}]})^{[\mathtt{i}]}, ((\mathbf{P}_{\mathtt{j}}^{[\mathtt{k}]})^{-1})^{[\mathtt{i}]})$$

$$\alpha = min\left\{\frac{p(\boldsymbol{\gamma}^{\star}|\cdot)q(\boldsymbol{\gamma}^{\star}, (\boldsymbol{\gamma}_{\mathtt{j}}^{[\mathtt{k}]})^{[\mathtt{i}]})}{p((\boldsymbol{\gamma}_{\mathtt{j}}^{[\mathtt{k}]})^{[\mathtt{i}]}|\cdot)q((\boldsymbol{\gamma}_{\mathtt{j}}^{[\mathtt{k}]})^{[\mathtt{i}]}, \boldsymbol{\gamma}^{\star})}, 1\right\}$$

$$(\gamma_{\mathtt{j}}^{[\mathtt{k}]})^{[\mathtt{i + 1}]} = \mathtt{if(accepted)}\ \boldsymbol{\gamma}^{\star}\ \mathtt{else}\ (\gamma_{\mathtt{j}}^{[\mathtt{k}]})^{[\mathtt{i}]}\text{; Update } \boldsymbol{\eta}^{[\mathtt{k}]}$$

$$a = rk(\mathbf{K}_{\mathtt{j}}^{[\mathtt{k}]})/2 + a_{\mathtt{j}}^{[\mathtt{k}]} \quad b = \frac{1}{2}((\boldsymbol{\gamma}_{\mathtt{j}}^{[\mathtt{k}]})^{[\mathtt{i + 1}]})'\mathbf{K}_{\mathtt{j}}^{[\mathtt{k}]}(\boldsymbol{\gamma}_{\mathtt{j}}^{[\mathtt{k}]})^{[\mathtt{i + 1}]} + b_{\mathtt{j}}^{[\mathtt{k}]}$$

$$(\tau_{\mathtt{j}}^{2[\mathtt{k}]})^{[\mathtt{i + 1}]}|\cdot \sim IG(a, b)$$

    }

  }

}

Distributional regression

Backfitting with smoothing parameter selection

Adapted selection algorithm of Belitz and Lang (2008).

$\texttt{while(eps >}\ \varepsilon\ \texttt{\& i < maxit) \{}$

$\quad\texttt{for(k in 1:K) \{}$

$\quad\quad\texttt{for(j in 1:p) \{}$

Compute $\mathbf{W}$ and $\mathbf{z}$

Optimize $\tau_{\mathtt{j}}^{2[\mathtt{k}]}$

$\texttt{objfun(tau2) \{}$

$$\gamma_{\mathtt{j}}^{[\mathtt{k}]} = ((\mathbf{Z}_{\mathtt{j}}^{[\mathtt{k}]})'\mathbf{W}\mathbf{Z}_{\mathtt{j}}^{[\mathtt{k}]} + \frac{1}{\mathtt{tau2}}\mathbf{K}_{\mathtt{j}}^{[\mathtt{k}]})^{-1}(\mathbf{Z}_{\mathtt{j}}^{[\mathtt{k}]})'(\mathbf{z} - \boldsymbol{\eta}_{\mathtt{-j}}^{[\mathtt{k}]})$$

$$\mathbf{f}_{\mathtt{j}}^{[\mathtt{k}]} = \mathbf{Z}_{\mathtt{j}}^{[\mathtt{k}]}\gamma_{\mathtt{j}}^{[\mathtt{k}]}$$

Return `IC` based on $\mathbf{f}_{\mathtt{j}}^{[\mathtt{k}]}$

$\texttt{\}}$

Update $\boldsymbol{\eta}^{[\mathtt{k}]}$

$\quad\quad\texttt{\}}$

$\quad\texttt{\}}$

Compute new `eps`

$\texttt{\}}$

Distributional regression

Model choice

To compare models across different response distributions and predictors we rely on quantile residuals and the DIC.

Quantile residuals are defined as

$$r_i = \Phi^{-1}(u_i)$$

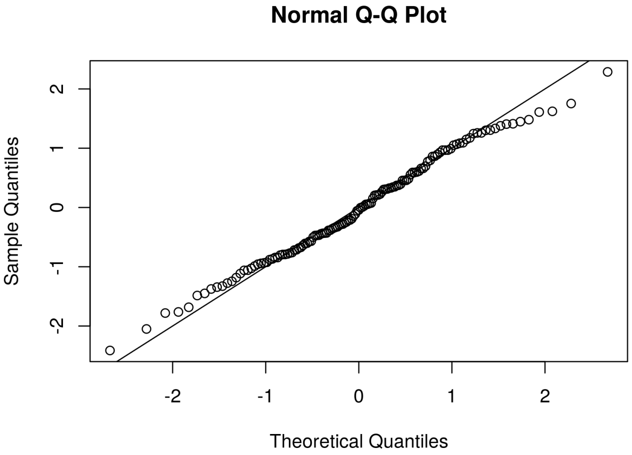
where Φ^{-1} is the cumulative distribution function of the standard normal distribution. From the cumulative distribution function of the response distribution obtain

$$u_i = \hat{F}(y_i)$$

for continuous responses. For discrete responses u_i is a random draw from the uniform distribution on the interval $\left[\hat{F}(y_i - 1), \hat{F}(y_i)\right]$.

Distributional regression
Model choice

Quantile residuals in the motorcycle example:



General architecture
Symbolic descriptions

Based on Wilkinson and Rogers (1973) a typical model description in R has the form

$$\texttt{response} \sim \texttt{x1 + x2}.$$

Using structured additive predictors we need generic descriptors for smooth/random terms, creating the type of term/basis we want to incorporate (model frame). The recommended R package **mgcv** (Wood 2006) has a pretty set up, e.g.

$$\texttt{response} \sim \texttt{x1 + x2 + s(z1) + s(z2, z3)}$$

$$\texttt{response} \sim \texttt{x1 + x2 + s(z1, bs = "ps")}.$$

General architecture
Symbolic descriptions

In the context of distributional regression we need formula extensions for multiple parameters. One convenient way to specify, e.g., the parameters of a normal model is:

```
list(
  response ~ x1 + x2 + s(z1) + s(z2),
  sigma ~ x1 + x2 + s(z1)
)
```

A four parameter example:

```
list(
  response ~ x1 + x2 + s(z1) + s(z2),
  sigma2 ~ x1 + x2 + s(z1),
  nu ~ s(z1),
  tau ~ s(z2)
)
```

General architecture
Symbolic descriptions

Hierarchical structures:

```
list(
  response ~ x1 + x2 + s(z1) + s(id1),
  id1 ~ x3 + s(z3) + s(id2),
  id2 ~ s(z4),
  sigma2 ~ x1 + x2 + s(z1),
  nu ~ s(z1) + s(id1),
  tau ~ s(z2)
)
```

Categorical responses:

```
list(
  response ~ x1 + x2 + s(z1) + s(z2),
  ~ x1 + x2 + s(z1) + s(z3)
)
```

General architecture
Symbolic descriptions

Hierarchical data set example:

	id1	x3	id2	z4
1	1	0.56	1	-0.49
2	2	1.36	1	-0.49
3	3	-0.78	1	-0.49
4	4	0.09	1	-0.49
5	5	-0.73	1	-0.49
6	1	0.56	2	-2.94
7	2	1.36	2	-2.94
8	3	-0.78	2	-2.94
9	4	0.09	2	-2.94
10	5	-0.73	2	-2.94

General architecture
Families

Families specify the details of models.

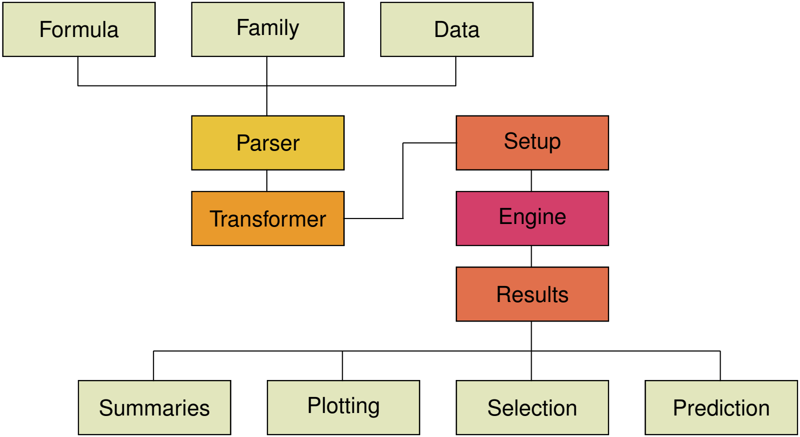
Required details may differ from engine to engine, however, to fully “understand” a distribution we need the following:

- The density function.
- The distribution function.
- The quantile function.
- Link function(s).
- A random number generator.
- First and second derivatives of the log-likelihood (expectations).

So implementing a “new” distribution means creating a new family (object), including the minimum specifications required by the estimating engine(s).

General architecture

Building blocks



In principle, the setup does not restrict to any specific type of engine (Bayesian or frequentist).

R package BayesR

The package is available at

https://R-Forge.R-project.org/projects/BayesR/

In R, simply type

```
R> install.packages("BayesR",
+    repos = "http://R-Forge.R-project.org")
```

R package BayesR
Available families

Work in progress . . . (+ note that not all families are available for all implemented engines yet)

BCCG	cloglog	lognormal	quant
beta	dagum	lognormal2	quant2
betazi	dirichlet	multinomial	t
betazi	gamma	mvn	truncgaussian
betazoi	gaussian	mvt	truncgaussian2
binomial	gaussian2	negbin	weibull
bivlogit	gengamma	pareto	zinb
bivprobit	invgaussian	poisson	zip

Families with ending 2 represent alternative parametrizations.

R package BayesR

Available building blocks

Type	Name
Parser	`parse.input.bayesr()`
Transformer	`randomize()`, `transformJAGS()`, `transformBayesX()`, `tranformIWLS()`
Setup	`setupJAGS()`, `jags2stan()`
Engine	`samplerBayesX()`, `samplerJAGS()`, `samplerSTAN()`, `samplerIWLS()`
Results	`resultsBayesX()`, `resultsJAGS()`, `resultsIWLS()`

R package BayesR

Input parameters

Parsing input parameters is based on **mgcv** infrastructures. In addition, the parser allows to define special user defined terms.

```
parse.input.bayesr(formula, data = NULL,
  family = gaussian.BayesR, weights = NULL,
  subset = NULL, offset = NULL, na.action = na.omit,
  contrasts = NULL, knots = NULL, specials = NULL,
  reference = NULL, ...)
```

Creates the model frame, all necessary matrices, to set up a model.

```
R> f <- list(accel ~ s(times), sigma ~ s(times))
R> pm <- parse.input.bayesr(f, data = mcycle)
R> names(pm)
```

```
[1] "mu"    "sigma"
```

```
R> names(pm$mu)
```

```
 [1] "formula"      "intercept"    "fake.formula" "response"
 [5] "pterms"       "sterms"       "smooth"       "sx.smooth"
 [9] "X"            "response.vec" "hlevel"
```

R package BayesR

Workflow example

JAGS

```
R> pm <- transformJAGS(pm)
R> ms <- setupJAGS(pm)
R> so <- samplerJAGS(ms)
R> mo <- resultsJAGS(pm, so)
R> summary(mo)
R> plot(mo)
```

BayesX

```
R> f <- list(
+   accel ~ sx(times),
+   sigma ~ sx(times)
+ )
R> pm <- parse.input.bayesr(f, data = mcycle)
R> pm <- transformBayesX(pm)
R> ms <- setupBayesX(pm)
R> so <- samplerBayesX(ms)
R> mo <- resultsBayesX(pm, so)
R> summary(mo)
R> plot(mo)
```

R package BayesR

Generic model fitting function

The “Lego” bricks are put together in the generic model fitting function `xreg()`, the main arguments are

```
xreg(formula, family = gaussian.BayesR, data = NULL,
  parse.input = parse.input.bayesr,
  transform = transformJAGS,
  setup = setupJAGS,
  engine = samplerJAGS,
  results = resultsJAGS,
  cores = NULL, combine = TRUE, model = TRUE, ...)
```

If new engines are implemented, one only needs to exchange the building block functions.

R package BayesR

Wrapper function

To ease the workflow, a wrapper function for the available engines is provided:

```
bayesr(formula, family = gaussian, data = NULL,
  knots = NULL, weights = NULL, subset = NULL,
  offset = NULL, na.action = na.fail, contrasts = NULL,
  engine = c("IWLS", "BayesX", "JAGS", "STAN"),
  cores = NULL, combine = TRUE,
  n.iter = 12000, thin = 10, burnin = 2000,
  seed = NULL, ...)
```

The function calls `xreg()` and returns an object of “`bayesr`” for which standard extractor and plotting functions are provided:

`summary()`, `plot()`, `fitted()`, `residuals()`, `predict()`, `coef()`, `DIC()`, `samples()`, `...`

Example

Munich rent data

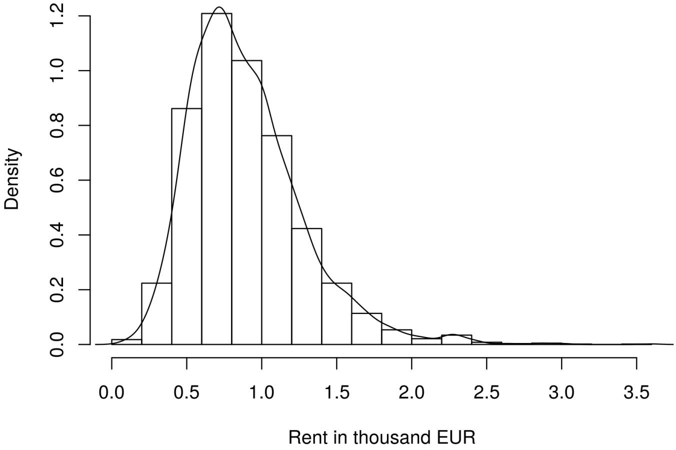
The aim is to establish a rent index to provide information on the "typical rent for a flat".

Variable	Description
`rent`	Net rent per month (EUR)
`rentsqm`	Net rent per month per square meter (EUR)
`area`	Living area in square meters
`yearc`	Year of construction
`location`	Quality of location: `"average"`, `"good"`, `"top"`
`bath`	Quality of the bathroom: `"standard"`, `"premium"`
`kitchen`	Quality of the kitchen: `"standard"`, `"premium"`
`cheating`	Central heating system: `"yes"`, `"no"`
`district`	District in Munich

Example

Munich rent data

```
R> data("rent99", package = "BayesR")
R> rent99$rent <- rent99$rent / 1000
```



Example
Munich rent data

Gaussian model in **BayesX**

```
R> data("MunichBnd", package = "BayesR")
R> f <- list(
+   rent ~ bath + kitchen + location + cheating +
+     sx(area) + sx(yearc) + sx(district, bs="mrf", map=MunichBnd),
+   sigma2 ~ bath + kitchen + location + cheating +
+     sx(area) + sx(yearc) + sx(district, bs="mrf", map=MunichBnd)
+ )
R> r1 <- bayesr(f, family = gaussian2,
+   data = rent99, engine = "BayesX")
R> summary(r1)
```

```
Call:
bayesr(formula = f, family = gaussian2, data = rent99,
    engine = "BayesX", verbose = TRUE)

Family: gaussian2
Link function: mu = identity, sigma2 = log
---
```

...continued on next slide...

Example

Munich rent data

```
Results for mu:
---
Formula:
rent ~ bath + kitchen + location + cheating + sx(area) +
    sx(yearc) + sx(district, bs = "mrf", map = MunichBnd)

Parametric coefficients:
                  Mean      Sd    2.5%     50% 97.5%
(Intercept)    0.86907 0.02326 0.82326 0.86922 0.916
bathpremium    0.07942 0.02286 0.03408 0.08019 0.122
kitchenpremium 0.10313 0.02505 0.05732 0.10311 0.152
locationgood   0.04988 0.01052 0.02934 0.04940 0.071
locationtop    0.14065 0.04301 0.05512 0.14132 0.225
cheatingyes    0.21268 0.01491 0.18225 0.21295 0.242

Smooth effects variances:
                  Mean        Sd      2.5%       50% 97.5%
sx(area)     0.0013374 0.0010928 0.0002956 0.0009991 0.004
sx(yearc)    0.0010686 0.0009757 0.0002476 0.0007958 0.004
sx(district) 0.0030714 0.0012751 0.0010150 0.0029200 0.006
```

Example

Munich rent data

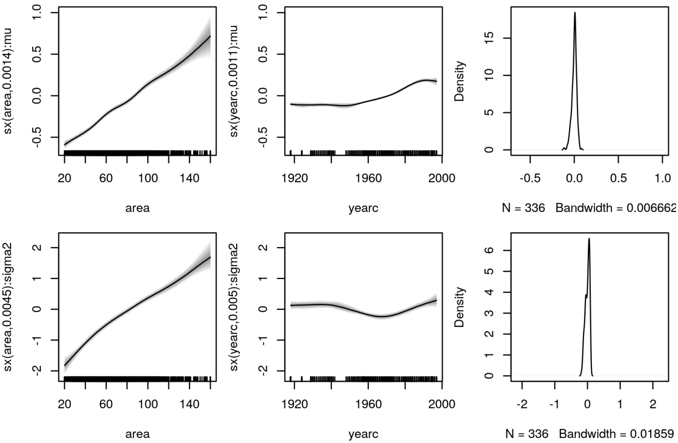
```
Results for sigma2:
---
Formula:
sigma2 ~ bath + kitchen + location + cheating + sx(area) +
    sx(yearc) + sx(district, bs = "mrf", map = MunichBnd)

Parametric coefficients:
                   Mean      Sd     2.5%      50%  97.5%
(Intercept)    -2.77689 0.10798 -2.99499 -2.77649 -2.565
bathpremium     0.17031 0.11550 -0.05608  0.17125  0.398
kitchenpremium  0.27878 0.13473  0.01762  0.27769  0.533
locationgood    0.27026 0.06359  0.14489  0.26974  0.397
locationtop     0.91968 0.17687  0.58473  0.91745  1.281
cheatingyes     0.12580 0.09313 -0.05675  0.12868  0.304

Smooth effects variances:
                 Mean        Sd      2.5%       50% 97.5%
sx(area)      0.0042278 0.0056634 0.0004687 0.0023609 0.019
sx(yearc)     0.0055512 0.0067109 0.0007356 0.0033074 0.021
sx(district)  0.0387951 0.0250549 0.0074447 0.0330498 0.099
```

Example
Munich rent data

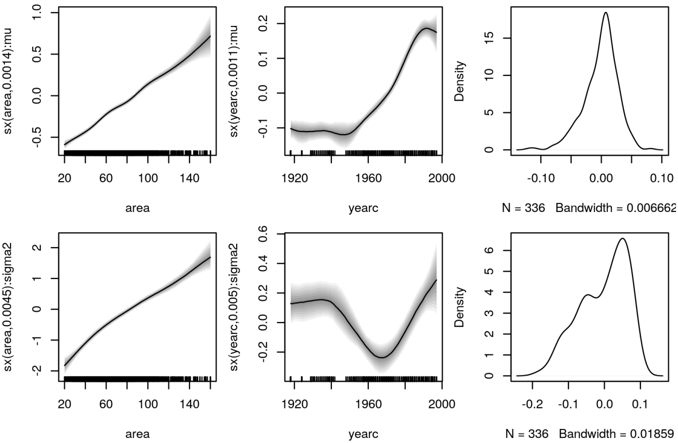
```
R> plot(r1, density = TRUE)
```



Example

Munich rent data

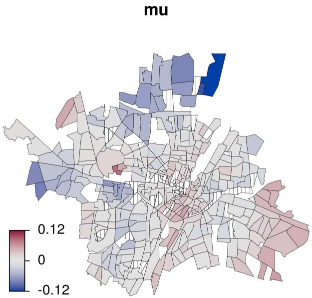
```
R> plot(r1, density = TRUE, scale = 0)
```

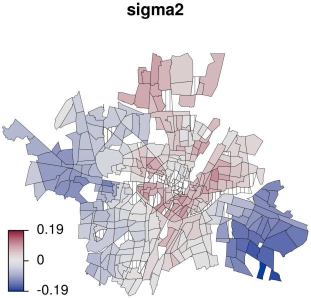


Example

Munich rent data

```
R> plot(r1, term = "sx(district)", map = MunichBnd)
```

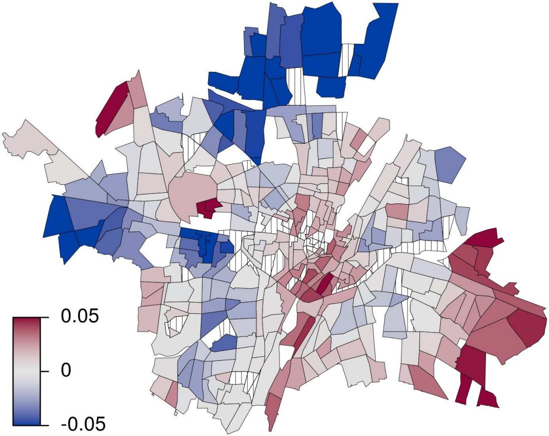




Example

Munich rent data

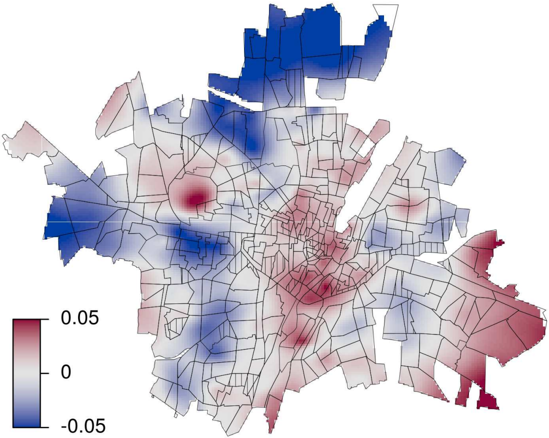
```
R> plot(r1, model = "mu", term = "sx(district)",
+   map = MunichBnd, range = c(-0.05, 0.05))
```



Example

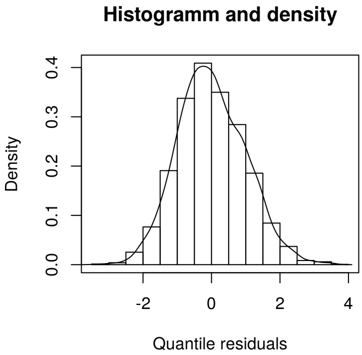
Munich rent data

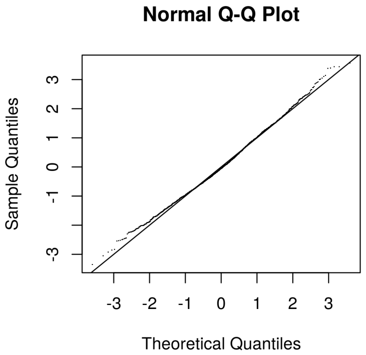
```
R> plot(r1, model = "mu", term = "sx(district)",
+   map = MunichBnd, range = c(-0.05, 0.05), interp = TRUE)
```



Example
Munich rent data

```
R> plot(r1, which = 3:6)
```

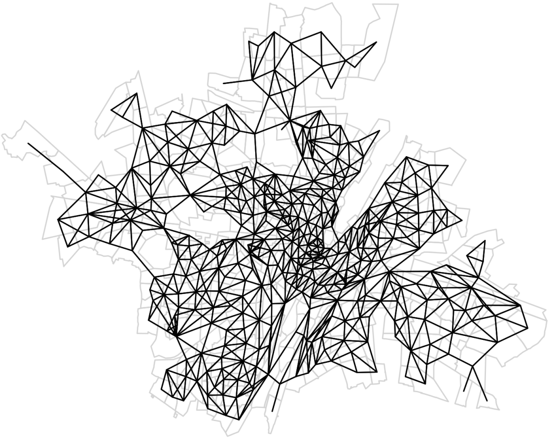




Example
Munich rent data

Neighborhood structures 1:

```
R> plotneighbors(MunichBnd, type = "boundary")
```

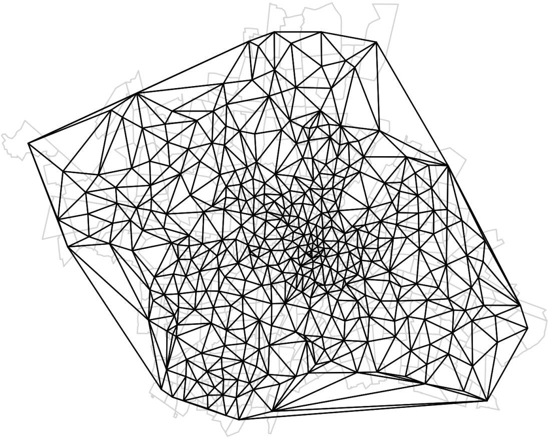


Example

Munich rent data

Neighborhood structures 2:

```
R> plotneighbors(MunichBnd, type = "delaunay")
```



Example

Munich rent data

Neighborhood structures 3:

```
R> plotneighbors(MunichBnd, type = "knear")
```

Example

Munich rent data

Neighborhood structures 3:

```
R> plotneighbors(MunichBnd, type = "knear", k = 2)
```

Example
Munich rent data

Creating a neighborhood structure to be used in **BayesX**

```
R> nm <- neighbormatrix(MunichBnd, type = "knear", k = 2)

R> f <- list(
+   rent ~ bath + kitchen + location + cheating +
+     sx(area) + sx(yearc) + sx(district, bs="mrf", map=nm),
+   sigma2 ~ bath + kitchen + location + cheating +
+     sx(area) + sx(yearc) + sx(district, bs="mrf", map=nm)
+ )

R> b <- bayesr(f, family = gaussian2,
+   data = rent99, engine = "BayesX")
```

Example
Munich rent data

Implementing the gamma distribution for **BayesX**.

```
gamma.BayesR <- function(...)
{
  rval <- list(
    "family" = "gamma",
    "names" = c("mu", "sigma"),
    "links" = c(mu = "log", sigma = "log"),
    "bayesx" = list(
      "mu" = c("gamma_mu", "mean"),
      "sigma" = c("gamma_sigma", "shape")
    ),
    "d" = function(y, eta, log = FALSE) {
      a <- exp(eta$sigma)
      s <- exp(eta$mu) / a
      dgamma(y, shape = a, scale = s, log = log)
    },
    "p" = function(y, eta, lower.tail = TRUE, log.p = FALSE) {
      a <- exp(eta$sigma)
      s <- exp(eta$mu) / a
      pgamma(y, shape = a, scale = s, lower.tail = lower.tail, log.p = log.p)
    }
  )

  rval
}
```

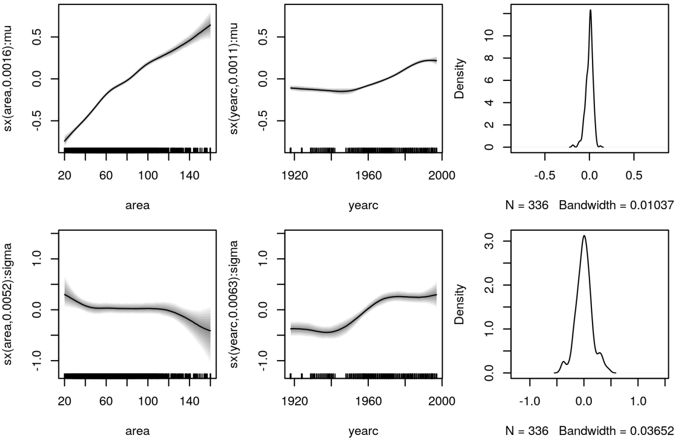
Example
Munich rent data

Gamma model in **BayesX**

```
R> f <- list(
+   rent ~ bath + kitchen + location + cheating +
+     sx(area) + sx(yearc) + sx(district, bs="mrf", map=MunichBnd),
+   sigma ~ bath + kitchen + location + cheating +
+     sx(area) + sx(yearc) + sx(district, bs="mrf", map=MunichBnd)
+ )
R> r2 <- bayesr(f, family = gamma,
+   data = rent99, engine = "BayesX")
```

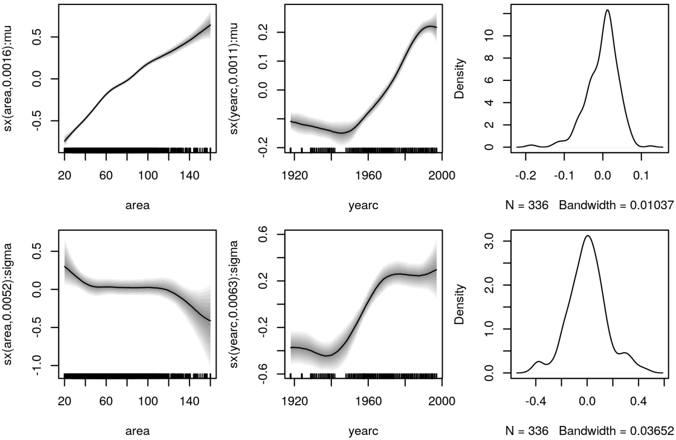
Example
Munich rent data

```
R> plot(r2, density = TRUE)
```



Example
Munich rent data

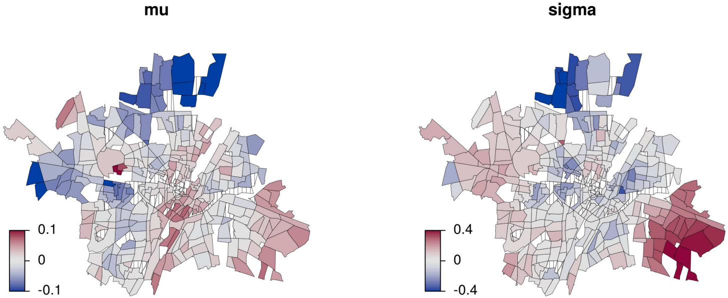
```
R> plot(r2, density = TRUE, scale = 0)
```



Example

Munich rent data

```
R> plot(r2, term = "sx(district)", map = MunichBnd)
```



Example

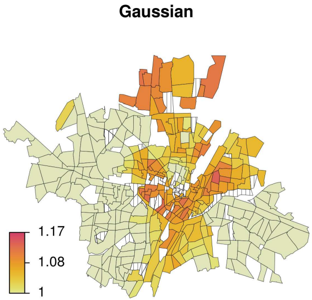
Munich rent data

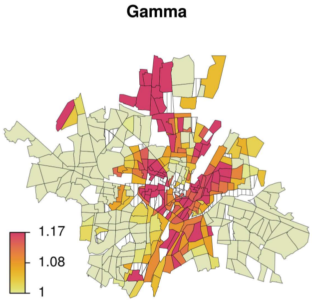
```
R> fsp1 <- fitted(r1, term = "sx(district)",
+   type = "parameter", samples = TRUE,
+   intercept = FALSE)
R> fsp2 <- fitted(r2, term = "sx(district)",
+   type = "parameter", samples = TRUE,
+   intercept = FALSE, FUN = function(x) { x })
R> sigma2 <- NULL
R> for(i in 1:ncol(fsp2$mu))
+   sigma2 <- cbind(sigma2, fsp2$mu[, i]^2 / fsp2$sigma[, i])
R> sigma2 <- apply(sigma2, 1, mean)

R> plotmap(MunichBnd, x = fsp1$sigma2, id = rent99$district,
+   col = heat_hcl, swap = TRUE, range = c(1, 1.17))
R> plotmap(MunichBnd, x = sigma2, id = rent99$district,
+   col = heat_hcl, swap = TRUE, range = c(1, 1.17))
```

Example

Munich rent data






```
R> DIC(r1, r2)
```

```
         DIC       pd
r1 -206.2633 100.4837
r2 -291.3151 132.5380
```

Example

Munich rent data

Implementing the gamma distribution for **IWLS**.

```
gamma.BayesR <- function(...)
{
  rval <- list(
    ...

    "score" = list(
      "mu" = function(y, eta, ...) {
        exp(eta$sigma) * (-1 + y / exp(eta$mu))
      },
      "sigma" = function(y, eta, ...) {
        mu <- exp(eta$mu)
        sigma <- exp(eta$sigma)
        sigma * (log(sigma) + 1 - log(mu) - digamma(sigma) + log(y) - y / mu)
      }
    ),
    "weights" = list(
      "mu" = function(y, eta, ...) { exp(eta$sigma) },
      "sigma" = function(y, eta, ...) {
        sigma <- exp(eta$sigma)
        sigma^2 * trigamma(sigma) - sigma
      }
    ),

    ...
  )

  rval
}
```

Example

Munich rent data

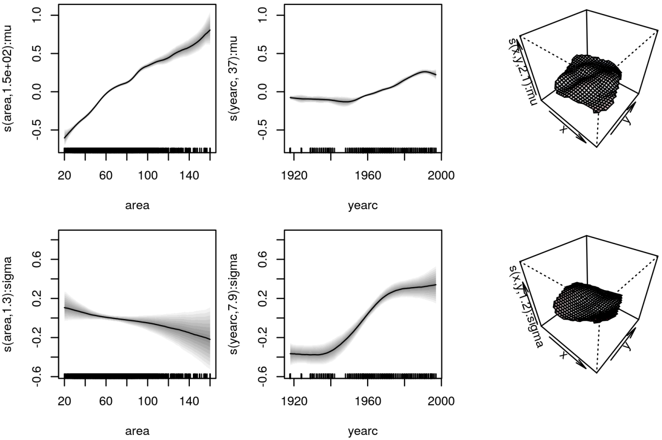
Gamma model with **IWLS**

```
R> rent99 <- cbind(rent99,
+   centroids(MunichBnd, id = rent99$district))
R> f <- list(
+   rent ~ bath + kitchen + location + cheating +
+     s(area) + s(yearc) + s(x, y, k = 100),
+   sigma ~ bath + kitchen + location + cheating +
+     s(area) + s(yearc) + s(x, y, k = 100)
+ )
R> r3 <- bayesr(f, family = gamma, data = rent99,
+   engine = "IWLS", method = c("backfitting", "MCMC"))
```

Example

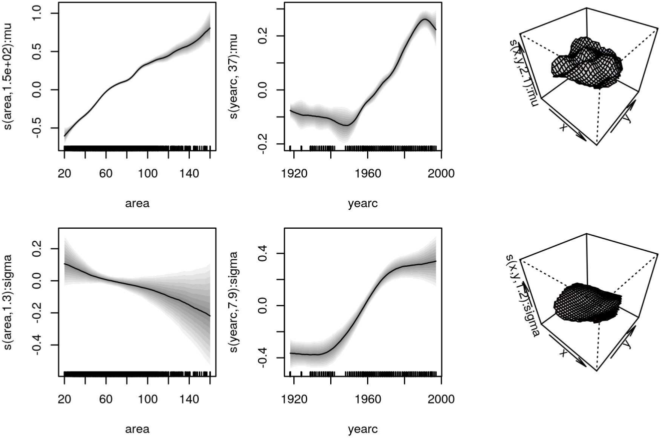
Munich rent data

```
R> plot(r3)
```



Example
Munich rent data

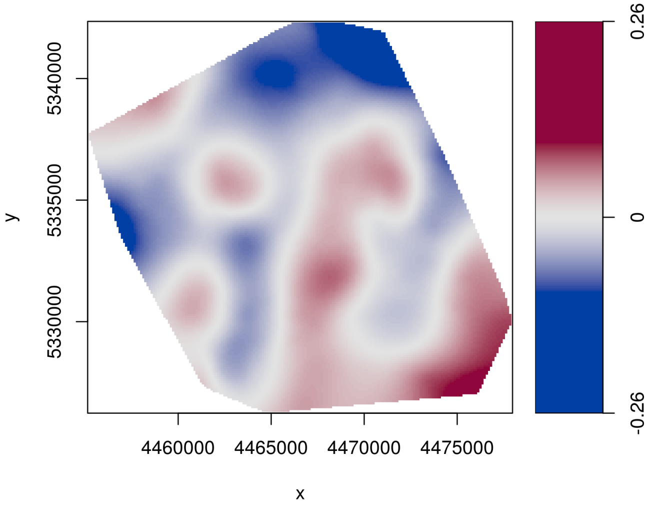
```
R> plot(r3, scale = 0)
```



Example

Munich rent data

```
R> plot(r3, model = "mu", term = "s(x,y)", image = TRUE)
```



Example

Munich rent data

Spatial predictions

```
R> grid <- 200
R> bbox <- bbox(bnd2sp(MunichBnd))
R> nd <- expand.grid(
+   "x" = seq(bbox["x", 1], bbox["x", 2], length = grid),
+   "y" = seq(bbox["y", 1], bbox["y", 2], length = grid)
+ )
```

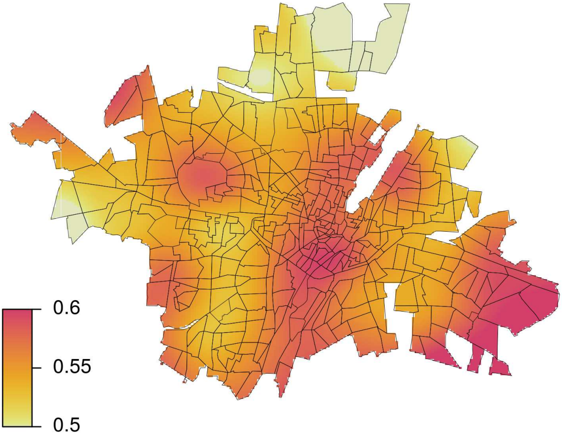
```
R> nd$fmu <- predict(r3, newdata = nd,
+   model = "mu", term = "s(x,y)",
+   type = "parameter")
```

```
R> i <- drop2poly(nd$x, nd$y, MunichBnd)
R> nd <- nd[i, ]
```

```
R> xymap(x, y, fmu, data = nd, col = heat_hcl,
+   symmetric = FALSE, swap = TRUE,
+   range = c(0.5, 0.6))
R> plotmap(MunichBnd, add = TRUE)
```

Example

Munich rent data



Thank you!!!

Belitz C, Brezger A, Kneib T, Lang S (2011). ***BayesX*** *– Software for Bayesian Inference in Structured Additive Regression. Models*. Version 2.0.1. URL `http://www.BayesX.org/`

Fahrmeir L, Kneib T, Lang S, Marx B (2013). *Regression – Models, Methods and Applications*. Springer, Berlin.

Klein N, Kneib T, Lang S (2013b). *Bayesian Structured Additive Distributional Regression*. Working Paper 2013-23, Working Papers in Economics and Statistics, Research Platform Empirical and Experimental Economics, Universität Innsbruck, August 2013. URL `http://econpapers.repec.org/paper/innwpaper/2013-23.htm`.

Umlauf N, Adler D, Kneib T, Lang S, Zeileis A (2012). *Structured additive regression models: An R interface to* ***BayesX***. Working Paper 2012-10, Working Papers in Economics and Statistics, Research Platform Empirical and Experimental Economics, Universität Innsbruck, May 2012. URL `http://CRAN.R-project.org/package=R2BayesX`

Rigby RA, Stasinopoulos DM (2005). *Generalized Additive Models for Location, Scale and Shape (with Discussion)*. Applied Statistics 54, 507–554.

Wood SN (2006). *Generalized Additive Models: An Introduction with R*. Chapman & Hall/CRC, Boca Raton.

Wood SN (2011). ***mgcv****: GAMs with GCV/AIC/REML Smoothness Estimation and GAMMs by PQL*. R package version 1.7-6. URL `http://CRAN.R-project.org/package=mgcv`