Distributional Modelling in R

Bayesian Distributional Regression

Thomas Kneib, Nikolaus Umlauf

https://nikum.org/dmr.html

Bayesian Inference

- Two central components of a Bayesian model formulation:
  - Observation model $f(\boldsymbol{y}|\vartheta)$ which describes how the data $\boldsymbol{y}$ are generated for given model parameters ϑ.
  - Prior distribution $f(\vartheta)$ representing prior beliefs about the parameter vector ϑ
- Bayesian learning updates prior beliefs on ϑ based on information in the data $\boldsymbol{y}$ using Bayes' theorem

$$f(\vartheta|\boldsymbol{y}) = \frac{f(\boldsymbol{y}|\vartheta)f(\vartheta)}{f(\boldsymbol{y})} = \frac{f(\boldsymbol{y}|\vartheta)f(\vartheta)}{\int f(\boldsymbol{y}|\vartheta)f(\vartheta)d\vartheta}$$

where $f(\boldsymbol{y})$ is the marginal density of the data.

Priof Beliefs and Prior Elicitation

- Main conceptual difference between likelihood-based and Bayesian inference: Coming up with a sensible prior distribution.
- The prior should reflect your prior beliefs about the parameter of interest.
- Very common practice:
  - Pick a mathematically convenient class of distributions for the prior and
  - only decide on the parameter of this prior distribution.
- For example, one can formulate belief statements such as

$$\mathbb{P}(c_1 \leq \vartheta \leq c_2) = 1 - \alpha$$

where c_1 and c_2 are pre-specified constants from which the prior parameters are determined.

Relation to Maximum Likelihood Estimation

- If the prior distribution is flat, i.e.

$$f(\vartheta) \propto \text{const},$$

  the posterior is proportional to the likelihood:

$$f(\vartheta|\boldsymbol{y}) = \frac{f(\boldsymbol{y}|\vartheta)f(\vartheta)}{f(\boldsymbol{y})} \propto f(\boldsymbol{y}|\vartheta)f(\vartheta) \propto f(\boldsymbol{y}|\vartheta).$$

- Hence the mode of the posterior coincides with the maximum likelihood estimate.
- In general,
  - the likelihood is a central part of Bayes' theorem that quantifies the information coming from the data and
  - the posterior forms a compromise between data (likelihood) and prior beliefs (prior).

Relation to Maximum Likelihood Estimation

- A typical discussion on Bayesian inference is that
  - frequentist inference assumes a true, fixed parameter value whereas
  - Bayesian inference assumes the parameter to be a random variable.
- This is, in general, misleading since the prior is merely used to reflect prior (un)certainty about the parameter of interest.

Bayesian Regression Models

- The observation model $f(\boldsymbol{y}|\vartheta)$ coincides with the likelihood from the frequentist perspective.
- We have to specify prior distributions for the regression coefficients (and potentially additional nuisance parameters).
- Inference typically relies on Markov chain Monte Carlo simulation techniques that yield samples from the posterior distribution.

Bayesian Perspective on GAMLSS

- The likelihood / observation model corresponds to the conditional distribution of the response given covariates, i.e.

$$f(y_i|\boldsymbol{x}_i) = f(y_i|\vartheta(\boldsymbol{x}_i)),$$

- For the basis coefficients of additive terms, the quadratic penalty

$$\text{pen}(\gamma) = \lambda\gamma'\boldsymbol{K}\gamma$$

  is replaced by a multivariate normal prior $\gamma \sim \text{N}(\boldsymbol{0}, \tau^2\boldsymbol{K}^-)$ with density

$$f(\gamma|\tau^2) \propto \left(\frac{1}{\tau^2}\right)^{0.5\,\text{rk}(\boldsymbol{K})} \exp\left(-\frac{1}{2\tau^2}\gamma'\boldsymbol{K}\gamma\right).$$

Priors in GAMLSS

- The prior $f(\gamma|\tau^2)$ encourages smoothness or shrinkage via the precision matrix $\boldsymbol{K}$.
- The impact of the prior is determined based on the variance parameter τ^2.
- In a Bayesian model formulation, we can specify hyperpriors on τ^2 and estimate it along with all the other model parameters.
- Since $\boldsymbol{K}$ often does not have full rank, the prior $\gamma \sim \mathrm{N}(\mathbf{0}, \tau^2\boldsymbol{K}^-)$ is partially improper, leading to some mathematical intricacies.

Numerically Assessing the Posterior

- The ultimate outcome of a Bayesian data analysis is the posterior, reflecting posterior beliefs about the parameter of interest.
- This is often reduced to point estimates, credible intervals, etc.
- Unfortunately, in most models of reasonable complexity, the posterior is not analytically accessible.
- In particular, the normalizing constant

$$f(\boldsymbol{y}) = \int f(\boldsymbol{y}|\vartheta) f(\vartheta) d\vartheta$$

  is unknown and for models of at least moderate complexity it can also not easily be numerically determined.

Numerically Assessing the Posterior

- If we could obtain random samples $\vartheta^{[t]}$, $t = 1, \ldots, T$ from the posterior, we could empirically estimate any quantity of interest at any desired level of precision:
    - Posterior expectations can be determined based on the law of large numbers via
    $$\frac{1}{T}\sum_{t=1}^{T} g(\vartheta^{[t]}) \to \mathbb{E}(g(\vartheta)|\boldsymbol{y}).$$
    - Similar statements exist for empirical quantiles.
    - Even the complete posterior could be estimated based on histograms or kernel density estimates.
- Markov chain Monte Carlo simulations are a way of simulating from the unknown and numerically untractable posterior!

Basic Principles of MCMC

- Generate a Markov chain that iteratively samples new values $\vartheta^{[t]}$ given current values $\vartheta^{[t-1]}$.
- The transition probabilities are chosen such that the Markov chain converges to the posterior as its stationarity distribution.
- Mathematical theory ensures convergence towards the stationary distribution in the limit, but in practice convergence has to be monitored appropriately.
- The convergence behaviour also depends on the starting values
  $\Rightarrow$ Remove burn in period.
- Samples from a Markov chain exhibit serial dependence that has to be accounted for
  $\Rightarrow$ thin out the Markov chain to achieve approximate independence.

Advantages of MCMC

- General advantages:
  - Access to the complete posterior distribution (including uncertainy quantification) without requiring asymptotic considerations.
  - Divide and conquer approach based on updating blocks of parameters separately allows handling very complex models having hundreds or thousands of parameters.
  - Modular representation of hierarchically formulated statistical models where certain parts of the model can be replaced without affecting the other model components
  - From the samples of the model parameters, we can determine not only inferences about these parameters themselves, but also inference for complex functionals of these parameters.
- Specifically for GAMLSS:
  - Seamless integration of determining the smoothness variance τ^2.
  - Inference for interpretable quantities derived from the fitted model.

Hyperpriors for the Smoothing Variances

- The (conjugate) default prior is $\tau^2 \sim \text{IG}(a, b)$ with $a = b = \epsilon$ and a small constant ϵ.
- Various alternative priors with theoretical advantages have been suggested in the literature (half normal, half Cauchy, uniform, scale-dependent, ...).
- Effect selection can be achieved with spike-and-slab-type prior structures that combine a discrete prior for effect selection with a continuous prior for shrinkage / smoothness.

Bayesian Information Criteria

- In principle, the same tools as in frequentist inference can be used for model choice and checking (quantile residuals, scoring rules, information criteria).
- When performing inference with MCMC, the deviance information criterion (DIC) and the widely applicable information criterion (WAIC) replace AIC/BIC.
- Both have the structure

$$\mathrm{IC} = 2\left(D_{\mathrm{IC}} + p_{\mathrm{IC}}\right)$$

  with different choices for the model fit D_{IC} and the model complexity p_{IC}.

The *bamlss* package.

- *bamlss* stands for Bayesian Additive Models for Location, Scale, and Shape (and Beyond).
- It provides a flexible framework for fitting Bayeian GAMLSS.
- It offers a wide range of modeling options, including smooth functions, spatial effects, and variable selection.
- Optimizer and sampler functions can be exchanged.
- Modular smooth term updating and proposal functions.

The *bamlss* package.

Example:

Load the Munich rent data.

```
R> library("bamlss")
R> library("gamlss.dist")
R> data("rent", package = "gamlss.data")
```

Try some continuous distributions, select the distribution with the lowest DIC.

```
R> families <- list(NO, GA, BCPE, BCCG, TF)
R> dic <- list()
R> for(fam in families) {
+    b <- bamlss(R ~ 1, data = rent, family = fam)
+    dic[[fam()$family[1]]] <- DIC(b)
+  }
R> dic <- do.call("rbind", dic)
```

The *bamlss* package.

```
R> dic <- dic[order(dic[, "DIC"], decreasing = TRUE), ]
R> print(dic)
          DIC       pd
NO   28972.89 2.030889
TF   28896.32 2.924092
BCPE 28625.46 4.484632
GA   28615.83 2.121942
BCCG 28613.53 2.917897
```

The *bamlss* package.

Model using the BCCG() family. Set up the formula.

```
R> f <- ~ s(Fl) + s(A) + loc + H
R> f <- rep(list(f), 3)
R> f[[1]] <- update(f[[1]], R ~ .)
R> print(f)
```

```
[[1]]
R ~ s(Fl) + s(A) + loc + H

[[2]]
~s(Fl) + s(A) + loc + H

[[3]]
~s(Fl) + s(A) + loc + H
```

Estimate model.

```
R> b <- bamlss(f, data = rent, family = BCCG,
+    n.iter = 12000, burnin = 2000, thin = 10)
```

The *bamlss* package.

Model summary and DIC.

```
R> summary(b)
```

```
Call:
bamlss(formula = f, family = BCCG, data = rent, n.iter = 12000,
    burnin = 2000, thin = 10)
---
Family: BCCG
Link function: mu = identity, sigma = log, nu = identity
*---
Formula mu:
---
R ~ s(Fl) + s(A) + loc + H
-
Parametric coefficients:
               Mean    2.5%     50%   97.5% parameters
(Intercept)  710.08  671.41  709.55  751.35      674.9
loc2         105.07   63.65  105.77  146.79      104.4
```

The *bamlss* package.

```
loc3          156.90  109.00  157.45  205.18      155.5
H1           -192.43 -222.26 -192.97 -162.22     -193.1
-
Acceptance probability:
        Mean   2.5%    50% 97.5%
alpha 0.9363 0.5251 0.9999     1
-
Smooth terms:
                 Mean      2.5%       50%     97.5% parameters
s(Fl).tau21 5.074e+03 1.447e-01 1.109e+02 4.647e+04  1.301e+05
s(Fl).alpha 9.806e-01 8.286e-01 1.000e+00 1.000e+00         NA
s(Fl).edf   1.609e+00 9.984e-01 1.100e+00 4.264e+00  5.672e+00
s(A).tau21  3.066e+05 4.891e+04 2.100e+05 1.113e+06  4.149e+05
s(A).alpha  9.351e-01 6.330e-01 1.000e+00 1.000e+00         NA
s(A).edf    4.903e+00 3.740e+00 4.846e+00 6.574e+00  5.535e+00
---
Formula sigma:
---
```

The *bamlss* package.

```
~s(Fl) + s(A) + loc + H
-
Parametric coefficients:
                 Mean        2.5%         50%       97.5% parameters
(Intercept) -0.9068524 -1.0192338 -0.9056053 -0.7896354     -0.762
loc2        -0.1226576 -0.2399974 -0.1235874 -0.0005593     -0.145
loc3        -0.1553606 -0.2922335 -0.1577257 -0.0274028     -0.145
H1           0.0934843  0.0024528  0.0938096  0.1847593      0.078
-
Acceptance probability:
        Mean   2.5%    50% 97.5%
alpha 0.9432 0.6621 0.9934     1
-
Smooth terms:
                 Mean      2.5%       50%     97.5% parameters
s(Fl).tau21 0.1962323 0.0002853 0.0758248 1.1215272      0.341
s(Fl).alpha 0.9378113 0.5955122 0.9954717 1.0000000         NA
s(Fl).edf   3.1104090 1.0447024 3.0144284 5.8896522      4.400
```

The *bamlss* package.

```
s(A).tau21  1.0157258 0.0004917 0.4200682 6.1329275      1.571
s(A).alpha  0.9301896 0.5923075 0.9991138 1.0000000         NA
s(A).edf    3.6919873 1.0585726 3.7916178 6.1307534      4.854
---
Formula nu:
---
~s(Fl) + s(A) + loc + H
-
Parametric coefficients:
                Mean     2.5%      50%    97.5% parameters
(Intercept)  0.56496  0.22888  0.56569  0.91684      0.939
loc2        -0.06440 -0.43149 -0.06385  0.28894     -0.288
loc3        -0.02758 -0.42061 -0.02986  0.37428     -0.099
H1          -0.31342 -0.55422 -0.31085 -0.07554     -0.226
-
Acceptance probability:
        Mean   2.5%    50% 97.5%
alpha 0.8820 0.3786 0.9839     1
```

The *bamlss* package.

```
-
Smooth terms:
                 Mean      2.5%       50%     97.5% parameters
s(Fl).tau21 8.631e-02 7.856e-05 6.297e-03 5.816e-01     11.943
s(Fl).alpha 9.537e-01 6.139e-01 1.000e+00 1.000e+00         NA
s(Fl).edf   1.365e+00 1.001e+00 1.112e+00 3.093e+00      6.748
s(A).tau21  1.731e-01 7.954e-05 8.118e-03 1.318e+00      0.002
s(A).alpha  9.498e-01 6.311e-01 9.970e-01 1.000e+00         NA
s(A).edf    1.449e+00 1.001e+00 1.143e+00 3.198e+00      1.129
---
Sampler summary:
-
DIC = 27653.84 logLik = -13811.29 pd = 31.2659
runtime = 215.225
---
Optimizer summary:
-
```

The *bamlss* package.

```
AICc = 27709.73 edf = 40.3372 logLik = -13813.67
logPost = -14081.54 nobs = 1969 runtime = 9.278
```

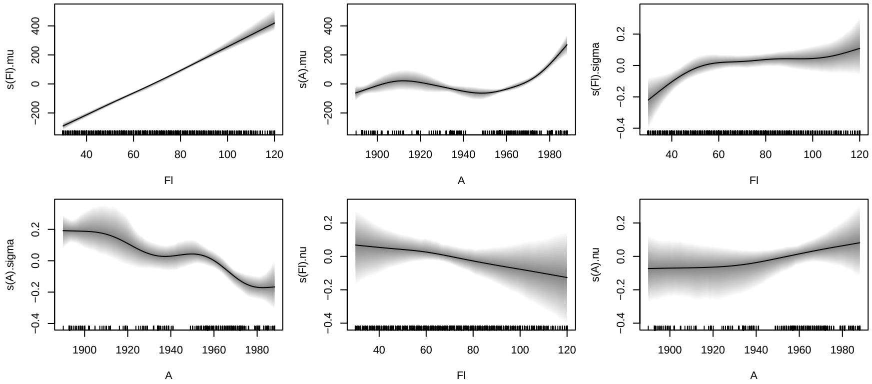
```
R> DIC(b)
```

```
      DIC       pd
 27653.84 31.26594
```

The *bamlss* package.

Plot estimated effects.

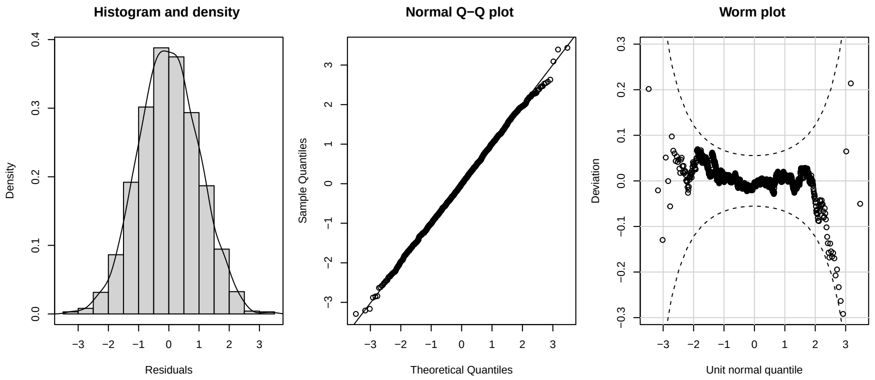
```
R> par(mfrow = c(2, 3), mar = c(4, 4, 1, 1))
R> plot(b, pages = 1, spar = FALSE, rug = TRUE)
```



The *bamlss* package.

Residual diagnostics.

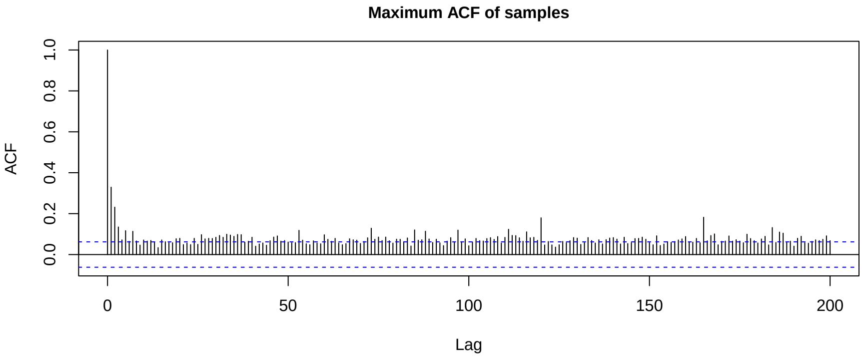
```
R> par(mfrow = c(1, 3), mar = c(4, 4, 4, 1))
R> plot(b, which = 3:5, spar = FALSE)
```



The *bamlss* package.

Maximum autocorrelation.

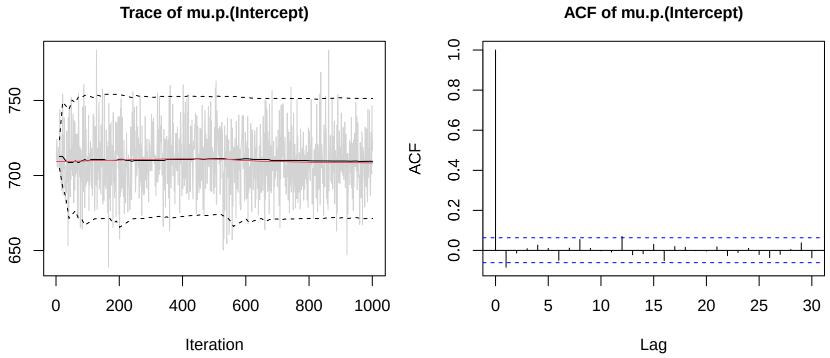
```
R> par(mar = c(4, 4, 4, 1))
R> plot(b, which = "max-acf", spar = FALSE, lag = 200)
```



The *bamlss* package.

Traceplots.

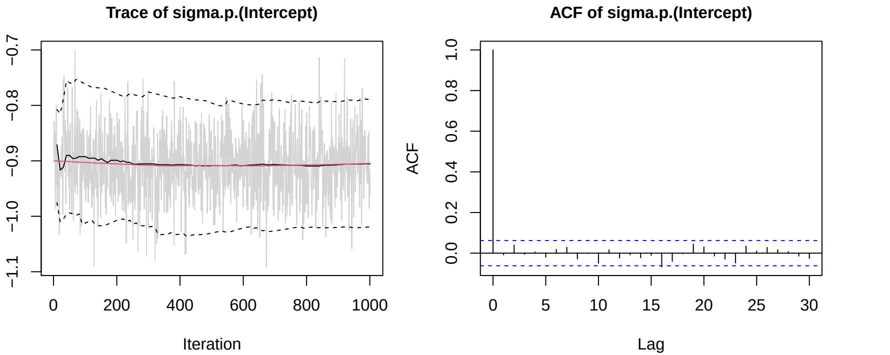
```
R> par(mar = c(4, 4, 4, 1))
R> plot(b, which = "samples", model = "mu", term = "(Intercept)")
```



The *bamlss* package.

Traceplots.

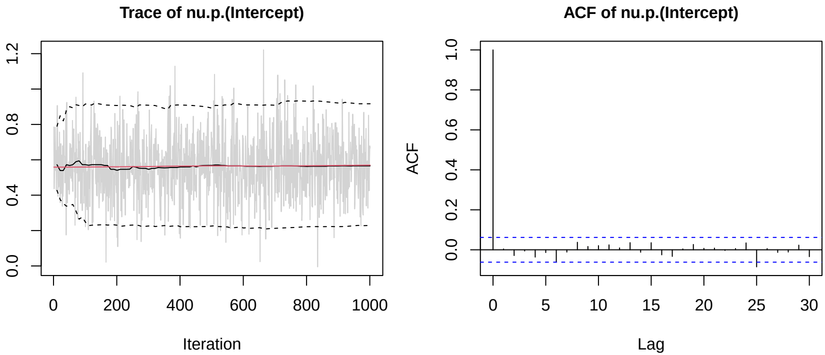
```
R> par(mar = c(4, 4, 4, 1))
R> plot(b, which = "samples", model = "sigma", term = "(Intercept)")
```



The *bamlss* package.

Traceplots.

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



The *bamlss* package.

Predictions.

```
R> nd <- rent[10, , drop = FALSE]
R> p <- predict(b, newdata = nd, model = "mu")
R> print(p)
```

```
[1] 789.4014
```

```
R> p <- predict(b, newdata = nd, type = "parameter")
R> print(p)
```

```
$mu
[1] 789.4014

$sigma
[1] 0.340004

$nu
[1] 0.5478488
```

The *bamlss* package.

```
R> p <- predict(b, newdata = nd, type = "parameter", FUN = median)
R> print(p)
```

```
$mu
[1] 789.7873

$sigma
[1] 0.3390889

$nu
[1] 0.5474406
```

```
R> p <- predict(b, newdata = nd, type = "parameter", model = "mu",
+     FUN = function(x) mean(x > 800))
R> print(p)
```

```
[1] 0.2377622
```

The *bamlss* package.

```
R> p <- predict(b, newdata = nd, type = "parameter", FUN = c95)
R> print(p)
$mu
       2.5%     Mean    97.5%
10 757.6416 789.4014 819.1297

$sigma
        2.5%     Mean     97.5%
10 0.3145338 0.340004 0.3682303

$nu
       2.5%      Mean     97.5%
10 0.401524 0.5478488 0.7211026
```