Hierarchische Modelle

Volker Schmid

19./26. Juni 2017

Hierarchische Modelle

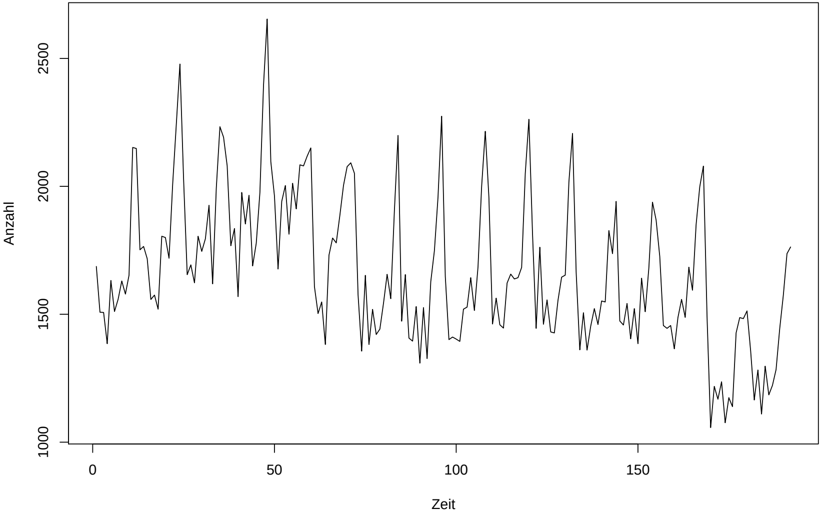
Hierarchische Modelle

Ein sogenanntes hierarchisches Modell umfasst mehrere Schichten, oft drei:

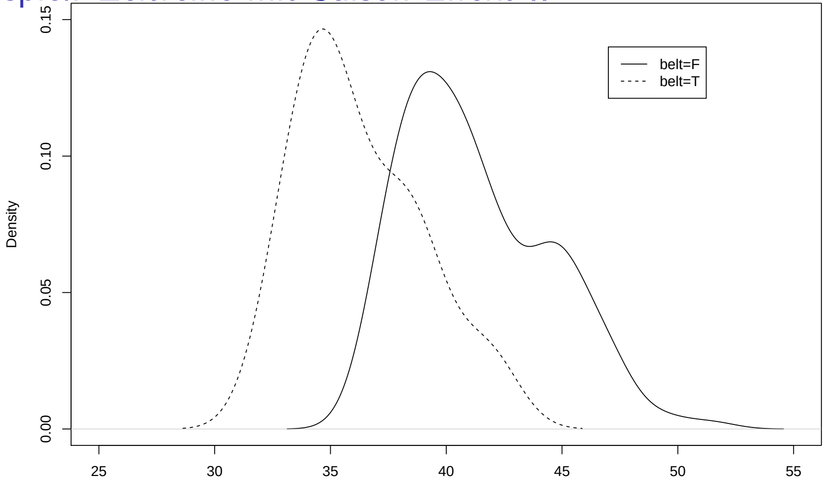
- Level 1: Datenmodell, Definition der Likelihood
- Level 2: Priori-Modell der unbekannten Parameter
- Level 3: (Hyper-)Prioris der Prioriparameter in Level 2

Beispiel: Zeitreihe mit Saison-Effekt I

- Anzahl von getöteten oder schwer verletzten Autofahrern in England von Januar 1969 bis Dezember 1984



Beispiel: Zeitreihe mit Saison-Effekt II



- Klarer Saisonaler Effekt
- Eventuell Trend über die Jahre
- Einfluß des Sicherheitsgurts

Datenmodell

- wir nehmen $\sqrt{y}$ als normalverteilt an

$$\sqrt{y_i} \sim N(\mu_i, \sigma^2); i = 1, \ldots, T = 192$$

- wir teilen den Erwartungswert μ_i in verschiedene Effekte auf:

$$\mu_i = \alpha + \beta x_i + \gamma_i + \delta_i$$

mit i Monat, x_i Dummyvariable Gurtpflicht ja/nein.

Priori-Modell - Zeittrend I

- Lineares Modell wäre vermutlich falsch
- Parametrische Modellierung eventuell möglich, aber welches Modell?
- Idee: Dummykodierung, ein Parameter pro Monat
- Aber: zu viele Parameter
- Idee: Aufeinander folgende Monate haben ähnliche Parameter
- Random Walk-Priori (1. Ordnung)

$$\gamma_i | \gamma_{i-1}, \tau_c \sim N(\gamma_{i-1}, \tau_c^{-1}); j = 2, \ldots, T;\ p(\gamma_1) \propto const.$$

Es gilt:

$$\gamma | \tau_c \sim N_T \left(0, (\tau_c Q_c)^{-1} \right)$$

Priori-Modell - Zeittrend II

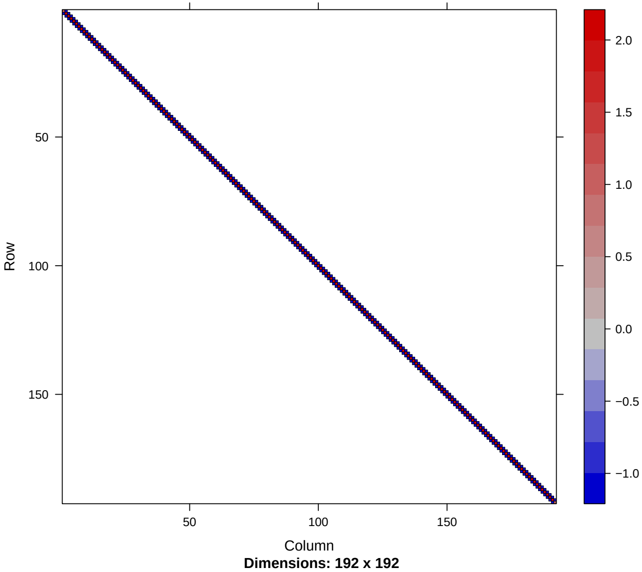
mit

$$Q_c = \begin{pmatrix} 1 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{pmatrix}$$

(fehlende Einträge sind 0)

```
library(Matrix)
Qc <- Matrix::sparseMatrix(i = c(1:T,1:(T-1)),
        j=c(1:T,2:T),
        x=c(1,rep(2,T-2),1,rep(-1,T-1)),
        symmetric = TRUE)
image(Qc)
```

Priori-Modell - Zeittrend III



Priorimodell - Saisontrend I

- Möglich wäre z.B. parametrische Sinus-Kurve, aber vermutlich zu unflexibel
- Sehr flexibel: Dummykodierung für jeden Monat. Aber: Viele Parameter, zu wenig Information
- Idee: Gleicher Monat im Folgenden Jahr hat ähnlichen Parameter. Die Summe über $m = 12$ Monate ist daher ähnlich (normal)verteilt. Formuliert als Priori-Information:

$$p(\delta|\tau_d) \propto (\tau_d^{T-m+1})/2 \cdot \\ \cdot \exp\left(-\frac{\tau_d}{2}\sum_{i=1}^{T-m+1}(\delta_i + \delta_{i+1} + \ldots + \delta_{i+m-1})^2\right)$$

Priorimodell - Saisontrend II

- Es lässt sich zeigen:

$$\delta|\tau_d \sim N_T\left(0, (\tau_d Q_d)^{-1}\right)$$

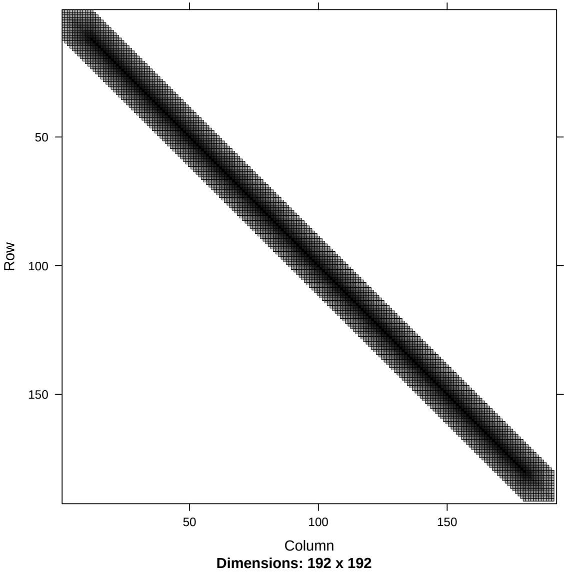
mit

$$Q_d = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 & & & & & \\ 1 & 2 & 2 & \cdots & 2 & 2 & 1 & & & & \\ 1 & 2 & 3 & \cdots & 3 & 3 & 2 & 1 & & & \\ \vdots & \vdots & \vdots & \ddots & & & & & & & \\ 1 & 2 & 3 & \cdots & 11 & 12 & 11 & 10 & \cdots & & \\ & 1 & 2 & 3 & \cdots & 11 & 12 & 11 & 10 & \cdots & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & \end{pmatrix}$$

Priorimodell - Saisontrend III

```
Qd <- Matrix::sparseMatrix(i=T,j=T,x=0,
        symmetric = TRUE)
EinsM<-Matrix::sparseMatrix(i=rep(1:12,12),
        j=rep(1:12,each=12), x=1)
for (i in 1:(T-12))
  Qd[i+(0:11),i+(0:11)]<-Qd[i+(0:11),i+(0:11)]+EinsM
image(Qd)
```

Priorimodell - Saisontrend IV



Priori-Modell - Intercept und Kovariableneffekt I

- Keine Priori-Information

$$p(\alpha) \propto const. \Leftrightarrow'' \alpha \sim N\left(0, 0^{-1}\right)''$$

- Analog $p(\beta) \propto const.$
- Also: Alle Effekte a priori normalverteilt, enthalten unbekannte Präzisionsparameter τ_c bzw. τ_d
- Gemeinsame Priori von $\theta = (\alpha, \beta, \gamma, \delta)$ hat Form

$$\theta|\tau \sim N_{2+2T}(0, Q(\tau)^{-1})$$
$$Q(\tau) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \tau_c Q_c & 0 \\ 0 & 0 & 0 & \tau_d Q_d \end{pmatrix}$$

Hyperpriors

- Die Präzisionsparameter in den Prioris steuern, wie stark die Regularisierung des Zeit- bzw. Saisontrends ist
- Diese Parameter der Prioris, *Hyperparameter*, wollen wir nicht vorgeben, sondern mitschätzen
- Jeder Parameter, der geschätzt werden soll, braucht eine Priori, hier *Hyperpriori* genannt

$$\tau_c \sim Ga(a_c, b_c)$$
$$\tau_d \sim Ga(a_d, b_d)$$

- Außerdem brauchen wir noch eine Priori für $\sigma^2 \sim IG(a_s, b_s)$

Posteriori

Sei $y^* = \sqrt{y}$

$$
\begin{aligned}
p(\theta, \tau | y^*) &\propto f(y^*|\theta) p(\theta, \tau) \\
&\propto f(y^*|\theta) p(\theta|\tau) p(\tau) \\
&\propto \prod_{i=1}^{T} \sigma^{-1} \exp\left(-\frac{1}{2\sigma^2} (y_i^* - \alpha - \beta x_i - \gamma_i + \delta_i)^2 \right) \cdot \\
&\cdot \tau_c^{(T-1)/2} \exp\left(-\frac{\tau_c}{2} \gamma^T Q_c \gamma \right) \tau_d^{(T-m+1)/2} \exp\left(-\frac{\tau_d}{2} \delta^T Q_d \delta \right) \\
&\cdot \tau_c^{a_c - 1} \exp(-\tau_c b_c) \cdot \tau_d^{a_d - 1} \exp(-\tau_d b_d) \\
&\cdot (\sigma^2)^{-a_s - 1} \exp(-b_s / \sigma^2)
\end{aligned}
$$

Full conditional - Zeitlicher Trend

$$
\begin{aligned}
p(\gamma|\cdot) &\propto \exp\left(-\frac{1}{2\sigma^2}\sum_i (y_i^* - \gamma_i - \delta_i - \beta x_i - \alpha)^2\right) \cdot \exp\left(-\frac{\tau_c}{2}\gamma^T Q_c \gamma\right) \\
&\propto \exp\left(-\frac{1}{2\sigma^2}\sum_i (\gamma_i - \epsilon_i^c)^2 - \frac{\tau_c}{2}\gamma^T Q_c \gamma\right) \\
&\propto \exp\left(-\frac{1}{2}\gamma^T \frac{T}{\sigma^2} I \gamma + \frac{1}{\sigma^2}\gamma^T \epsilon_c^* - \frac{1}{2}\gamma^T \tau_c Q_c \gamma\right) \\
&\propto \exp\left(-\frac{1}{2}\gamma^T \left(\frac{T}{\sigma^2} I + \tau_c Q_c\right)\gamma + \frac{1}{\sigma^2}\gamma^T \epsilon_c^*\right)
\end{aligned}
$$

Kanonische Form der Normalverteilung mit Präzisionsmatrix $Q_c^* = \left(\frac{1}{\sigma^2} I + \tau_c Q_c\right)$ und Erwartungswert $Q_c^{*-1} m_c$ mit $m_{c,i} = \frac{1}{\sigma^2}\sum_i (y_i^* - \alpha - \beta x_i - \delta_i)$.

Full conditional - Saison-Effekt

Analog:

$$
\begin{aligned}
\delta|\cdot &\sim N_T((Q_d^*)^{-1} m_d, (Q_d^*)^{-1}) \\
Q_d^* &= \left(\frac{1}{\sigma^2} I + \tau_d Q_d\right) \\
m_{d,k} &= \frac{1}{\sigma^2} \sum_i (y_i^* - \alpha - \beta x_i - \gamma_i)
\end{aligned}
$$

Full conditional - Kovariableneffekt

$$p(\beta|\cdot) \propto \exp\left(-\frac{1}{2\sigma^2}\sum_i (y_i^* - \gamma - \delta - \alpha - \beta x_i)^2\right)$$
$$\propto \exp\left(-\frac{1}{2}\beta^2 \frac{1}{\sigma^2}\sum_i x_i^2 + \beta\frac{1}{\sigma^2}\sum_i (x_i y_i^* - x_i\alpha - x_i\gamma_i - x_i\delta_i)\right)$$

Also $\beta|\cdot \sim N(q_b^{-1}m_b, q_b^{-1})$ mit $q_b = \frac{\sum x_i^2}{\sigma^2}$ und
$m_b = \frac{1}{\sigma 2}\sum_i (x_i y_i^* - x_i\alpha - x_i\gamma_j - x_i\delta_k)$

- Analog: $\alpha|\cdot \sim N(q_a^{-1}m_a, q_a^{-1})$ mit $q_a = \frac{T}{\sigma^2}$ und
  $m_b = \frac{1}{\sigma 2}\sum_i (y_i^* - x_i\beta - \gamma_j - \delta_k)$

Full Conditional - Präzisionsparameter**

Allgemein: $p(\tau|\cdot) \propto p(\theta|\tau)p(\tau)$, also ist die Full Conditional unabhängig von den Daten

Konkret:

$$\begin{aligned} p(\tau_c|\cdot) &\propto \tau_c^{(T-1)/2} \exp\left(\frac{\tau_c}{2}\gamma^T Q_c \gamma\right) \tau_c^{a_c-1} \exp(-\tau_c b_c) \\ &\propto \tau_c^{c+(T-1)/2-1} \exp\left(-\tau_c(b_c + \gamma^T Q_c \gamma/2)\right) \end{aligned}$$

Also

$$\begin{aligned} \tau_c|\cdot &\sim Ga\left(a_c + (T-1)/2, b_c + \gamma^T Q_c \gamma/2\right) \\ \tau_d|\cdot &\sim Ga\left(a_d + (T-m+1)/2, b_d + \delta^T Q_d \delta/2\right) \\ \sigma_2|\cdot &\sim IG\left(a_s + T/2, b_s + \sum (y_i^* - \alpha - x_i\beta + \gamma_j - \delta_k)^2\right) \end{aligned}$$

Implementation I

```
y <- sqrt(Drivers$y)
belt <- Drivers$belt

a.c <- a.d <- 1
b.c <- 0.0005
b.d <- 0.1
a.s <- 1/4
b.s <- 1/4

alpha <- mean(y)
beta <- 0
gamma <- rep(0,T)
delta <- rep(0,T)
tau.c <- a.c/b.c
tau.d <- a.d/b.d
sigma2 <- b.s/a.s
```

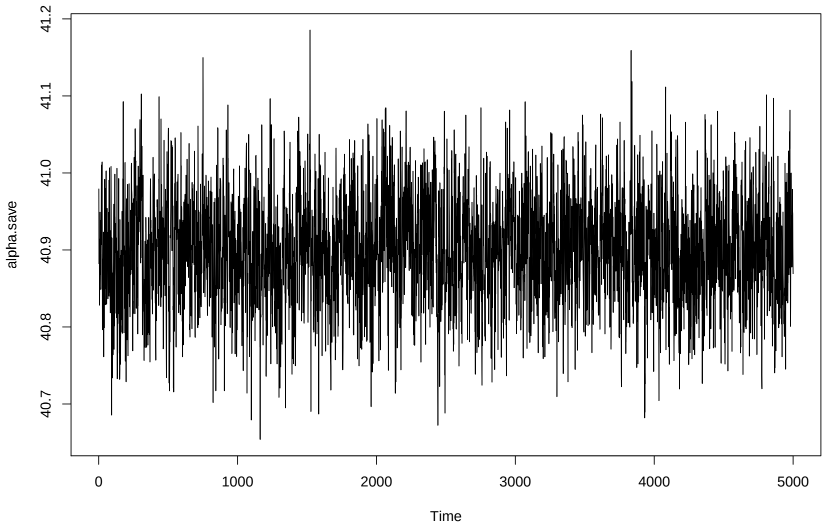
Implementation II

```
sumx2 <- sum(belt)
burnin=1000
nr.it=5000
I = burnin+nr.it

alpha.save<-beta.save<-sigma2.save<-rep(0,nr.it)
gamma.save<-array(0,c(T,nr.it))
delta.save<-array(0,c(T,nr.it))
tau.save<-array(0,c(2,nr.it))
source("../R/rmvnorm.R") #rmvnormcanon()
iter=0
```

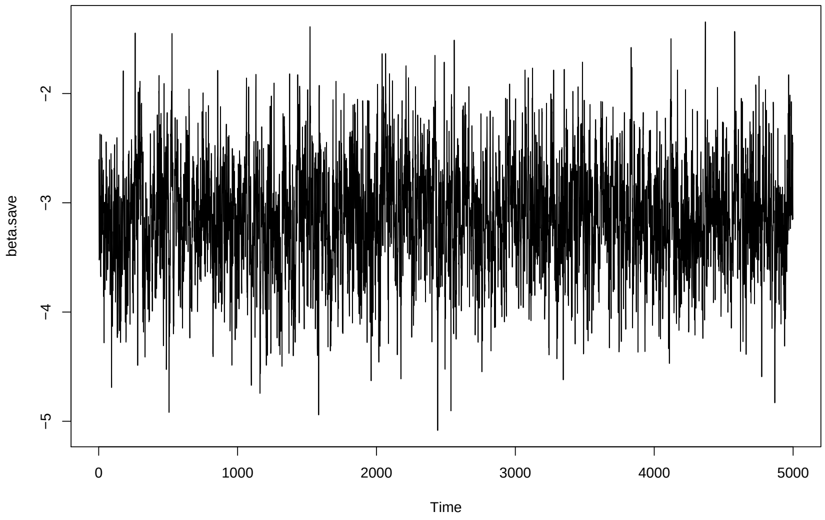
MCMC

```
while (iter<I)
{
  iter=iter+1
  gamma<-gamma-mean(gamma);  delta<-delta-mean(delta)
  m <- sum(y-gamma-delta-belt*gamma)/sigma2; Q <- T/sigma2
  alpha <- rnorm(1, m/Q, 1/Q)
  m <- sum(belt*(y-gamma-delta-alpha))/sigma2; Q <- sumx2/s
  beta <- rnorm(1, m/Q, 1/Q)
  m <- (y-delta-beta*belt-alpha)/sigma2; Q <- tau.c*Qc + Ma
  gamma <- rmvnormcanon(m,Q)
  m <- (y-gamma-beta*belt-alpha)/sigma2; Q <- tau.d*Qd + Ma
  delta <- rmvnormcanon(m,Q)
  tau.c <- rgamma(1, a.c + (T-1)/2, b.c + (t(gamma)%*%Qc%*%
  tau.d <- rgamma(1, a.d + (T-12+1)/2, b.d + (t(delta)%*%Qd
  sigma2 <- 1/rgamma(1, a.s + T/2, b.s + sum((y-alpha-gamma

  if (iter>burnin){alpha.save[iter-burnin]<-alpha;  beta.sa
    gamma.save[,iter-burnin]<-gamma; delta.save[,iter-burni
```

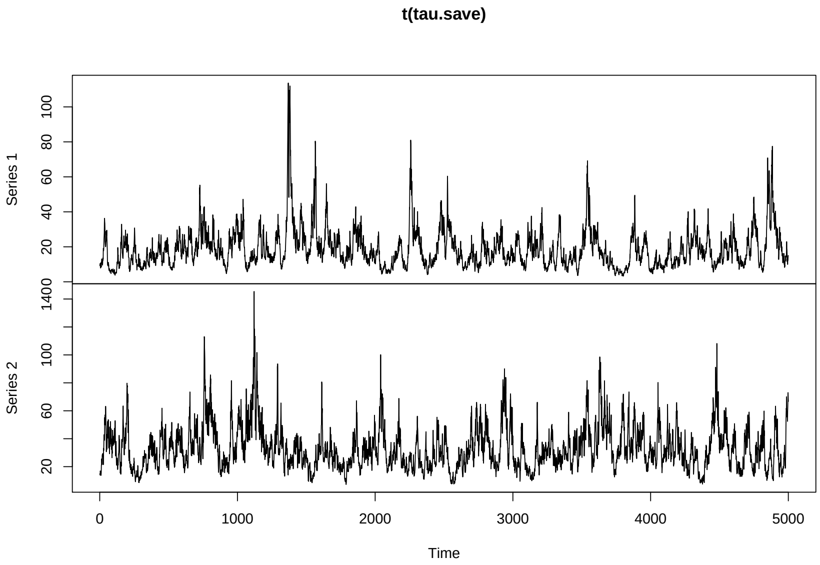
Ergebnisse I



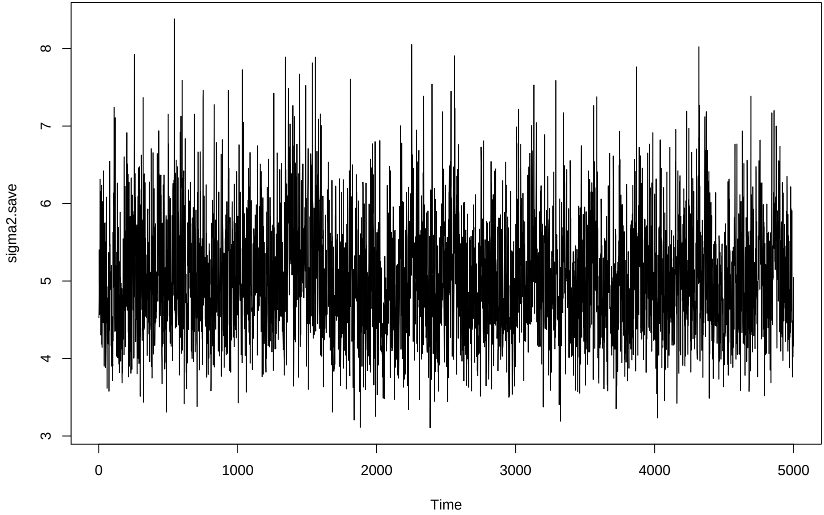
Ergebnisse II



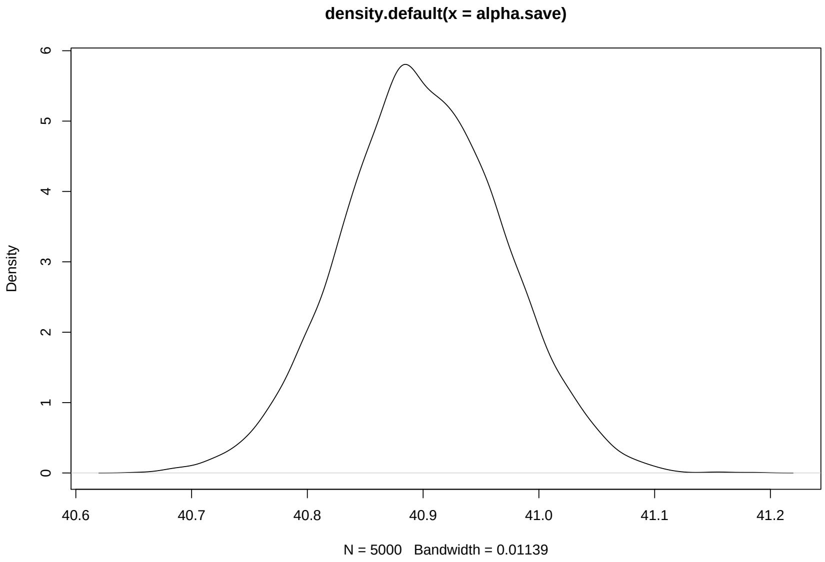
Ergebnisse III



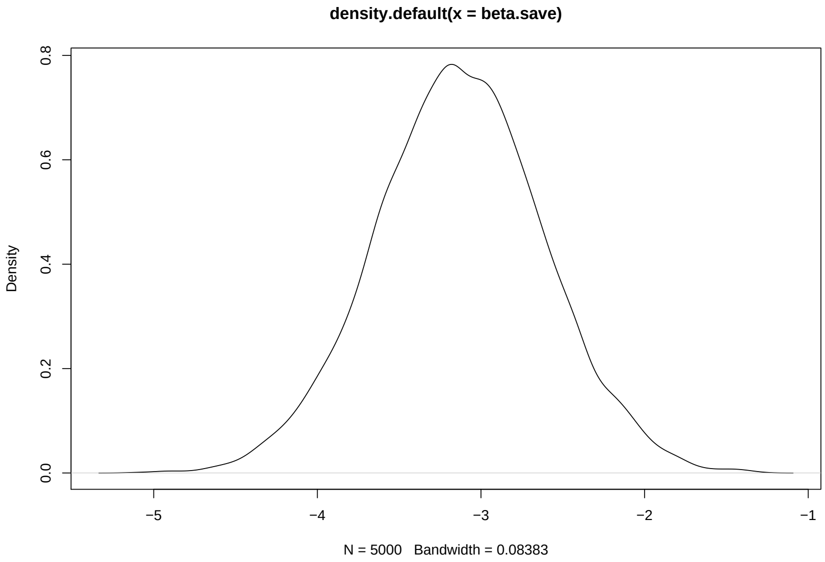
Ergebnisse IV



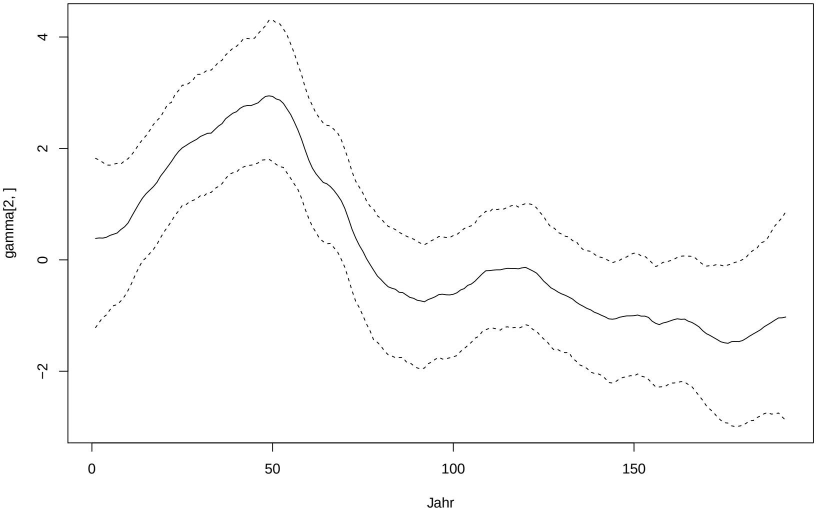
Ergebnisse V



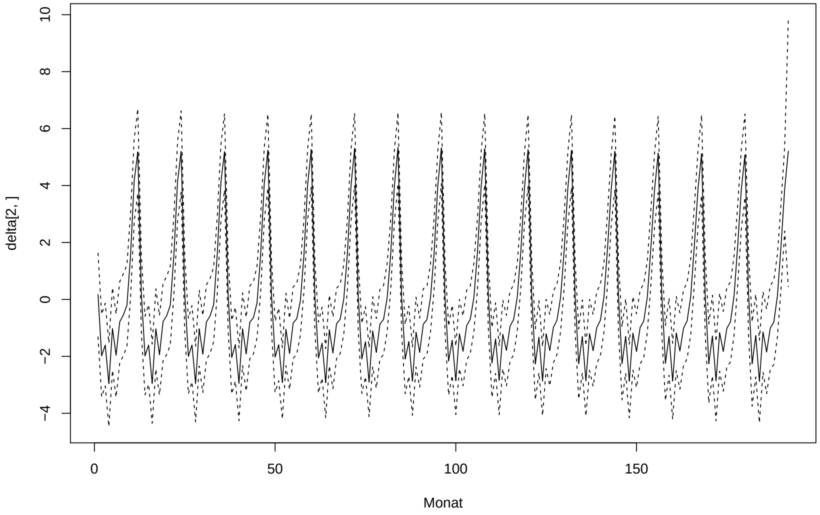
Ergebnisse VI



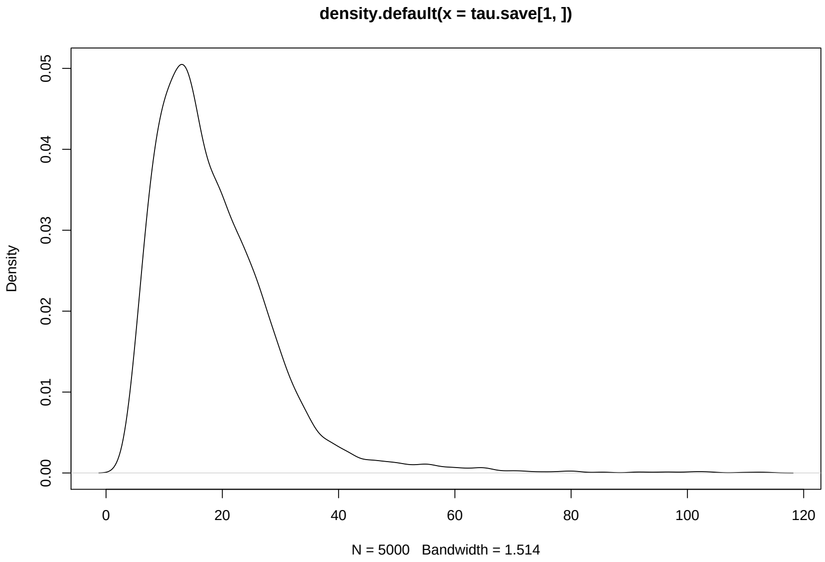
Ergebnisse VII



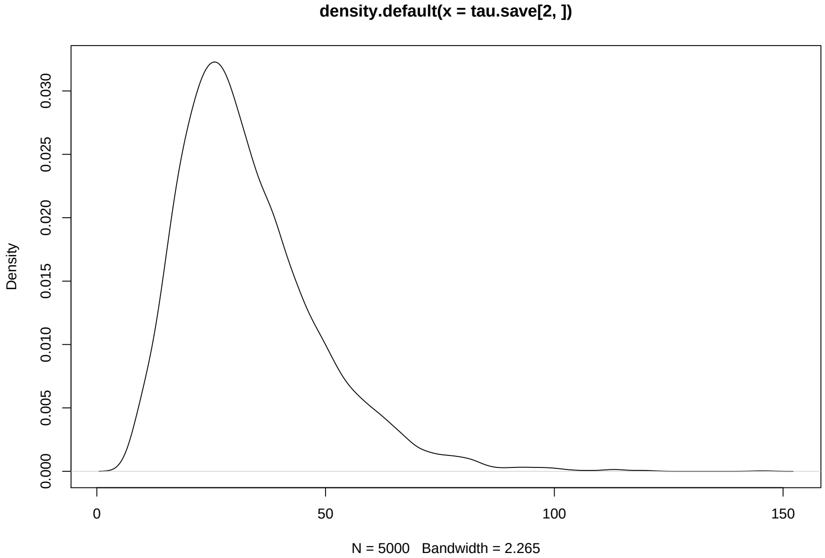
Ergebnisse VIII



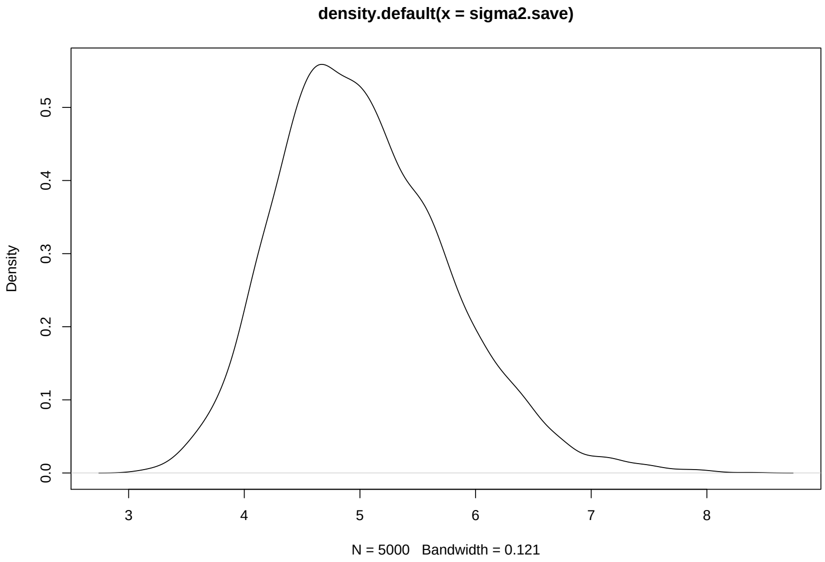
Ergebnisse IX



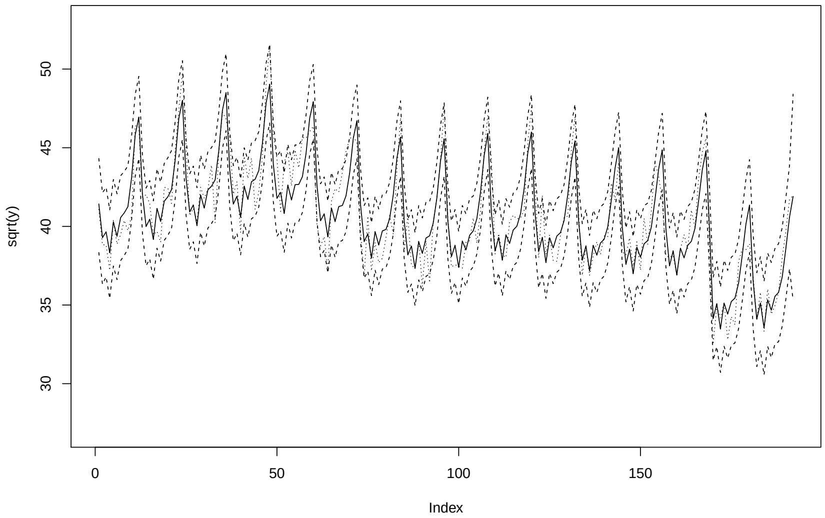
Ergebnisse X



Ergebnisse XI



Ergebnisse XII



BayesX I

*BayesX is a software tool for estimating **structured additive regression** models. Structured additive regression embraces several well-known regression models such as generalized additive models (GAM), generalized additive mixed models (GAMM), generalized geoadditive mixed models (GGAMM), dynamic models, varying coefficient models, and geographically weighted regression within a unifying framework...*

BayesX II

Figure 1: BayesX Logo

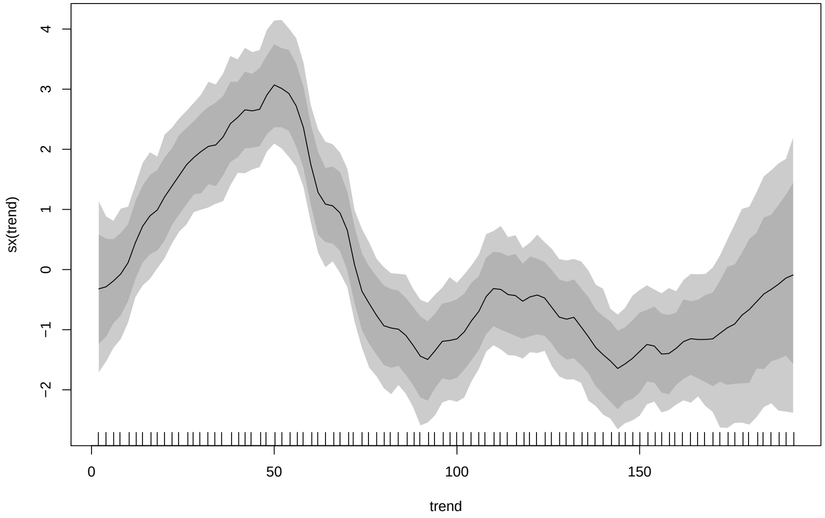
BayesX III

```r
#install.packages(c("BayesXsrc","BayesX"), type = "source")
library(R2BayesX)
#bayesx.term.options(bs = "rw1")
formula.bx  = sqrt(y) ~ belt +
  sx(trend, bs="rw1", a=1, b=0.0005) +
  sx(seasonal, bs="season", a=1, b=0.1)

system.time(
  mcmc.bx <- bayesx(formula = formula.bx, data=Drivers,
      control=bayesx.control(family="gaussian",
      method="MCMC", hyp.prior = c(4,4),
      iterations=10000L, burnin=5000L))
)
```

```
##    user  system elapsed
##   3.483   1.522   5.174
```

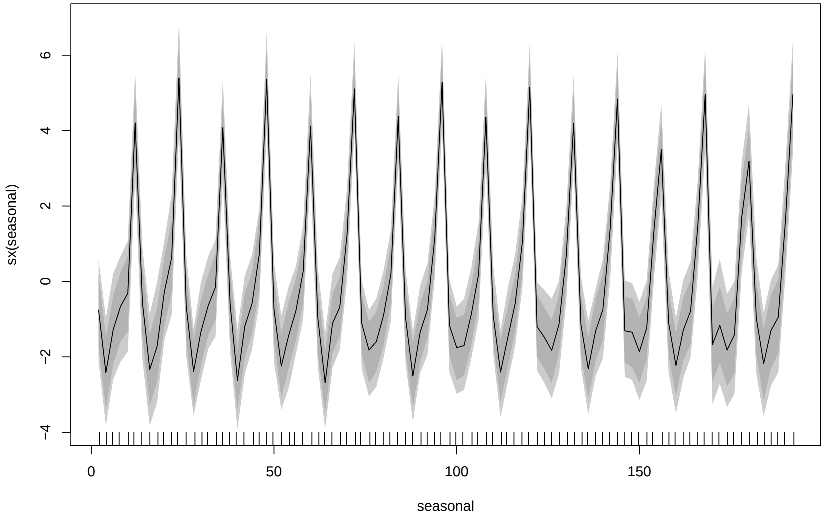
BayesX IV

```
plot(mcmc.bx$effects$`sx(trend)`)
```



BayesX V

```r
plot(mcmc.bx$effects$`sx(seasonal)`)
```



BayesX VI

```r
summary(mcmc.bx)
```

```
## Call:
## bayesx(formula = formula.bx, data = Drivers, control = b
##     method = "MCMC", hyp.prior = c(4, 4), iterations = 1
##     burnin = 5000L))
##
## Fixed effects estimation results:
##
## Parametric coefficients:
##                 Mean      Sd    2.5%     50%   97.5%
## (Intercept) 41.2577  0.1678 40.9269 41.2636 41.5819
## belt        -4.5165  1.0051 -6.4356 -4.4899 -2.5181
##
## Smooth terms variances:
##                 Mean     Sd    2.5%    50%  97.5%    Min
```

BayesX VII

```
## sx(seasonal) 0.2386 0.0853 0.1213 0.2217 0.4884 0.0728 0
## sx(trend)    0.1894 0.0836 0.0754 0.1725 0.3926 0.0536 0
##
## Scale estimate:
##          Mean     Sd   2.5%    50%  97.5%
## Sigma2 2.7923 0.3216 2.2100 2.7635 3.4787
##
## N = 192  burnin = 5000  DIC = 209.6865  pd = 45.12971
## method = MCMC  family = gaussian  iterations = 10000  st
```

Empirischer Bayes

Andere Ansätze

- Schön hier: Gibbs-Sampler, da wir semi-konjugierte Prioris verwenden
- Aber noch einfacher: wenn wir τ kennen würden, wäre $\theta|\cdot \sim N(.)$ und wir könnten Schlüsse analytisch ziehen
- Analogie zum penalisierten Likelihod-Ansatz:

$$\begin{aligned}\log(p(\theta|x)) &= \log(f(x|\theta)) + \log(p(\theta)) + C \\ &= l(\theta) + \frac{\tau}{2}\theta^T Q\theta + C = l_{pen}(\theta) + C\end{aligned}$$

Empirischer Bayes-Ansatz

- Beim empirischen Bayes-Ansatz wird keine Priori-Information für die Hyperparameter spezifiziert
- Die Hyperparameter werden aus den Daten geschätzt
- Kein Bayesianisches Verfahren im eigentlichen Sinn
- Verschiedene Methoden möglich

Expectation-Maximization-Algorithmus

Im obigen Beispiel wäre folgender Algorithmus sinnvoll, um den Maximum-A-Posteriori-Schätzer zu berechnen:

1. Schätze θ aus y bei gegebenen τ
2. Schätze τ als inverse Varianz aus θ

- Iteriere bis zur Konvergenz
- Empirischer Bayes-Ansatz würde nach weniger Iterationen abbrechen (z.B. 1. $\to$ 2. $\to$ 1.)

Restringierter ML-Schätzer (REML)

- Grundidee des REML: Transformiere Daten, so dass nuisance Parameter in der Likelihood nicht mehr auftauchen
- Hier: $y \sim N(X\theta, \sigma^2 I)$
- Transformation z.B. $A = I - X(X^T X)^{-1} X^T$:

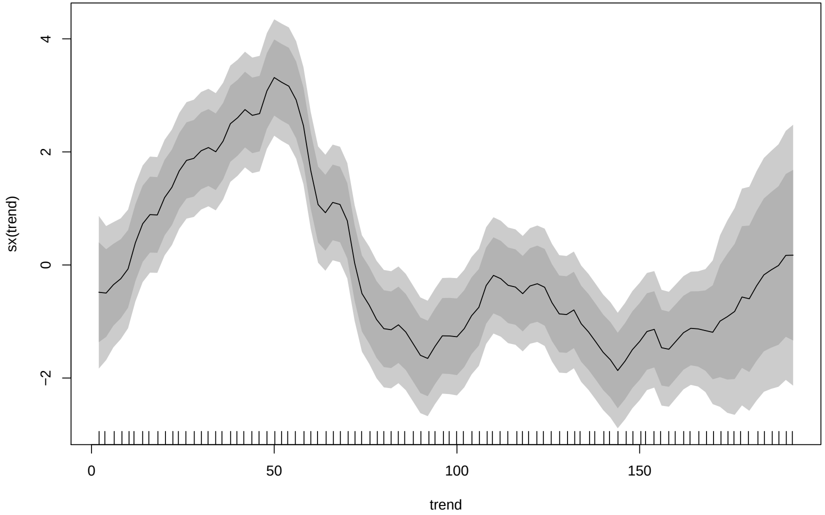
$$
\begin{aligned}
E(Ay) &= E((I - X(X^T X)^{-1} X^T)y) \\
&= X\theta - X(X^T X)^{-1} X^T X\theta = 0 \\
Var(Ay) &= (I - X(X^T X)^{-1} X^T)^T (\sigma^2 I)(I - X(X^T X)^{-1} X^T) \\
&= \sigma^2 (I - 2(X(X^T X)^{-1} X^T) \\
&+ X(X^T X)^{-1} X^T X(X^T X)^{-1} X^T) \\
&= \sigma^2 (I - X(X^T X)^{-1} X^T)
\end{aligned}
$$

- Schätze τ aus $p(\tau|Ay)$, hängt nicht mehr von θ ab
- Setze Schätzung in ursprüngliche Posteriori ein: $p(\theta|y, \hat{\tau})$

BayesX mit REML I

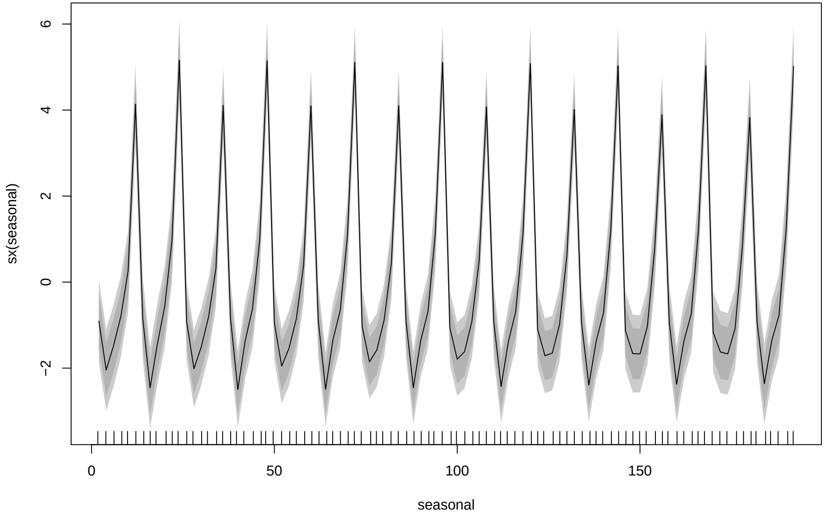
```r
# Keine Hyperparameter (!)
formula.bx2 = sqrt(y) ~ belt + sx(trend, bs="rw1") +
  sx(seasonal, bs="season")

reml.bx <- bayesx(formula = formula.bx2, data=Drivers,
              control=bayesx.control(family="gaussian",
              method="REML"))

plot(reml.bx$effects$`sx(trend)`)
```

BayesX mit REML II




```
plot(reml.bx$effects$`sx(seasonal)`)
```

BayesX mit REML III




```
summary(reml.bx)
```

BayesX mit REML IV

```
## Call:
## bayesx(formula = formula.bx2, data = Drivers, control =
##     method = "REML"))
##
## Fixed effects estimation results:
##
## Parametric coefficients:
##             Estimate Std. Error  t value  Pr(>|t|)
## (Intercept)  41.2769     0.1665 247.8527 < 2.2e-16 ***
## belt         -4.6514     1.0544  -4.4114 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.
##
## Smooth terms:
##               Variance Smooth Par.      df Stopped
## sx(seasonal)    0.0193    117.0120 14.2552       0
## sx(trend)       0.2541      8.8647 19.0758       0
```

BayesX mit REML V

```
##
## Scale estimate: 2.2527
##
## N = 192  df = 35.331  AIC = 383.257  BIC = 498.347
## GCV = 2.76067  logLik = -156.2975  method = REML  family
```

Model Averaging

- Uns interessiert eigentlich die marginale Posterioriverteilung von θ. Für diese gilt:

$$p(\theta|y) = \int p(\theta, \tau|y) d\tau = \int p(\theta|\tau, y) p(\tau|y) d\tau$$

- Im Gegensatz zum Empirischen Bayes-Ansatz erhalten wir beim vollen Bayes-Ansatz also nicht das Ergebnis für einen τ-Wert, sondern die Mischung von verschiedenen Modellen, gewichtet mit der marginalen Posteriori-Verteilung von τ.
- Man spricht vom *Model Averaging*

Integrated Nested Laplace Approximation

Integrated Nested Laplace Approximation (INLA) I

$$p(\tau|y) = \frac{p(\theta, \tau|y)}{p(\theta|\tau, y)}$$

- Zähler ist bekannt bis auf Normierungskonstante, Zähler ist Dichte der Normalverteilung.
- Für jeden gegebenen Wert von τ kann Posteriori-Modus $\theta^*(\tau)$ leicht berechnet werden
- Benutze Laplace-Approximation:

$$\tilde{p}(\tau|y) = \left. \frac{p(\theta, \tau|y)}{p(\theta|\tau, y)} \right|_{\theta=\theta^*(\tau)}$$

INLA-Algorithmus:

- Schätze Modus von $\tau|y$
- Berechne $\tilde{p}(\tau|y)$ an geschickt gewählten Stützpunken um den Modus (berücksichtige Hessematrix)
- Interpolation von $\tilde{p}(\tau|y)$ zwischen den Stützpunkten
- Schliesslich numerische Integration:

$$\tilde{p}(\theta|y) = \sum_{\tau} p(\theta|\tau, y)\tilde{p}(\tau|y)$$

R-INLA

```r
#install.packages("INLA", repos="https://inla.r-inla-downl
library(INLA)
formula  = sqrt(y) ~ belt + f(trend, model="rw1",
  param=c(a.c,b.c)) + f(seasonal, model="seasonal",
  season.length=12, param=c(a.d,b.d))
inla.mod <- inla(formula, family="gaussian",
  data=Drivers, control.family=list(param=c(a.s,b.s)))
plot(inla.mod)
summary(inla.mod)
```

```
##
## Call:
## c("inla(formula = formula, family = \"gaussian\", data =
##
## Time used:
##  Pre-processing    Running inla Post-processing
##          2.8367          2.7317          0.3517
##
```

Hierarchische Bayes-Modelle

Allgemeiner (oder “Generalisiert”“)

Allgemein gilt für hierarchische Modelle:

$$
\begin{aligned}
p(\theta|y) &= \int p(\theta|\tau, y) p(\tau|y)\; d\tau \\
&= \int \frac{p(y|\theta)p(\theta|\tau)}{p(y|\tau)} p(\tau|y)\; d\tau
\end{aligned}
$$

$$
p(\tau|y) = \int p(\tau|\theta) p(\theta|y)\; d\theta
$$

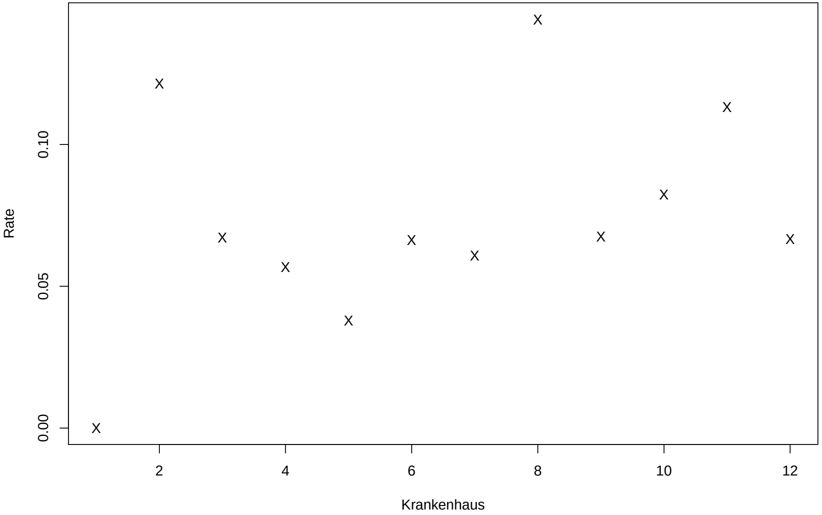
Beispiel: Modell mit zufälligen Effekten

Daten:

- Todesfälle nach Herzoperationen bei Babies
- Beobachtungen von 12 Krankenhäusern
- Annahme: Todesrate ist Krankenhausspezifisch
- Individuelle Raten der Krankenhäuser kommen aus gemeinamer Verteilung

Daten

```
library(INLA); data(Surg)
plot(Surg$r/Surg$n, pch='X', ylab='Rate', xlab='Krankenhaus
```



Hierarchisches Modell I

- Datenmodell: Sei y_i die Anzahl der Todesfälle bei n_i Operationen in Krankenhau $i = 1, \ldots, p = 12$;

$$y_i \sim B(n_i, \pi_i); i = 1, \ldots, p$$
$$\pi_i = \frac{exp(\theta_i)}{1 - \exp(\theta_i)}$$

- Priori-Modell

$$\theta_i|\tau \sim N\left(\mu, \tau^{-1}\right)\theta \quad \sim N_p\left(\mu, (\tau I)^{-1})\right)$$

- Hyper-Priori: Wie oben $\tau Ga(a, b)$, $a = b = 1/1000$.

Posteriori und Full Conditionals

- Unterschied zu oben: Likelihood ist nicht mehr normal
- Full Conditional von θ ist *fast* normal
- Full Conditional von τ ist aber unverändert
- Likelihood pro Krankenhaus:

$$
\begin{aligned}
f(y_i|\theta_i) &\propto \pi^{y_i}(1-\pi)^{n_i-y_i} \\
&\propto \exp(\theta_i)^{y_i}(1-\exp(\theta))^{-y_i}(1-\exp(\theta))^{-n_i+y_i} \\
&\propto \exp(y_i\theta_i)(1-\exp(\theta))^{n_i}
\end{aligned}
$$

Approximation der Full Conditionals I

Full Conditional:

$$p(\theta|\tau, y) \propto \prod_i \exp(y_i\theta_i)(1-\exp(\theta))^{n_i}\exp(-\frac{\tau}{2}((\theta-\mu)^T(\theta-\mu)))$$
$$\propto \exp\sum_i \left(y_i\theta_i - \frac{\tau}{2}\theta_i^2 + \tau(\theta_i-\mu) + n_i\log(1-\exp(\theta_i))\right)$$

Sei

$$\begin{aligned}
f(\theta) &:= \log(1-\exp(\theta)) \\
f(\theta)' &= \frac{-\exp(\theta)}{(1-\exp(\theta))} \\
f(\theta)'' &= \frac{-\exp(\theta)(1-\exp(\theta)) - (-\exp(\theta))(-\exp(\theta))}{(1-exp(\theta))^2} \\
&= \frac{-\exp(\theta)}{(1-\exp(\theta))^2}
\end{aligned}$$

Approximation der Full Conditionals II

Taylor-Approximation:

$$
\begin{aligned}
f(\theta_i) &\approx f(\theta_i^0) - (\theta_i - \theta_i^0) f'(\theta_i^0) + \frac{1}{2}(\theta_i - \theta_i^0)^2 f''(\theta_i^0) \\
&= \theta_i \left(f'(\theta_i^0) - \theta_i^0 f''(\theta_i^0) \right) + \frac{1}{2}\theta_i^2 f''(\theta_i^0) + const.
\end{aligned}
$$

Damit Approximation der Posteriori:

$$
\begin{aligned}
\tilde{p}(\theta|\tau, y) &\propto \exp\left(y_i\theta_i - \frac{\tau}{2}(\theta^T\theta) + \tau\mu\theta + \right. \\
&\qquad \left. + n_i \left(\theta_i \left(f'(\theta_i^0) - \theta_i^0 f''(\theta_i^0) \right) + \frac{1}{2}\theta_i^2 f''(\theta_i^0) \right) \right) \\
\tilde{p}(\theta_i|\tau, y) &\propto \exp\left(\theta_i(\mu + y_i + n_i f'(\theta_i^0) - n_i\theta_i^0 f''(\theta_i^0)) \right. \\
&\qquad \left. - \frac{1}{2}\theta_i^2(\tau + f''(\theta_i^0)) \right)
\end{aligned}
$$

Anwendung der Approximation I

- Bei **Markov Chain Monte Carlo**

1. benutze $\tilde{p}(\theta^{(k-1)})$ als Proposal, wobei $\theta^{(k-1)}$ der aktuelle Wert ist
2. benutze $\tilde{p}(\theta^*$ als Proposal, wobei θ^* (approximativ) der Modus von $p(\theta|\tau^{(k-1)})$ ist (IWLS-Proposal)

BayesX hat bei Kombination MCMC, Binomial und Zufälliger Effekt (random effect, re) einen bekannten Bug.

Anwendung der Approximation II

```r
library(R2BayesX)
SurgBx <- data.frame(y=as.factor(c(rep(1,sum(Surg$r)),
   rep(0,sum(Surg$n-Surg$r)))), hospital=c(
    rep(Surg$hospital,Surg$r),rep(Surg$hospital,
                                   (Surg$n-Surg$r))))
b <- bayesx(y ~ sx(hospital, bs = "re"),
    data = SurgBx, family = "binomial", method = "REML")
summary(b)
```

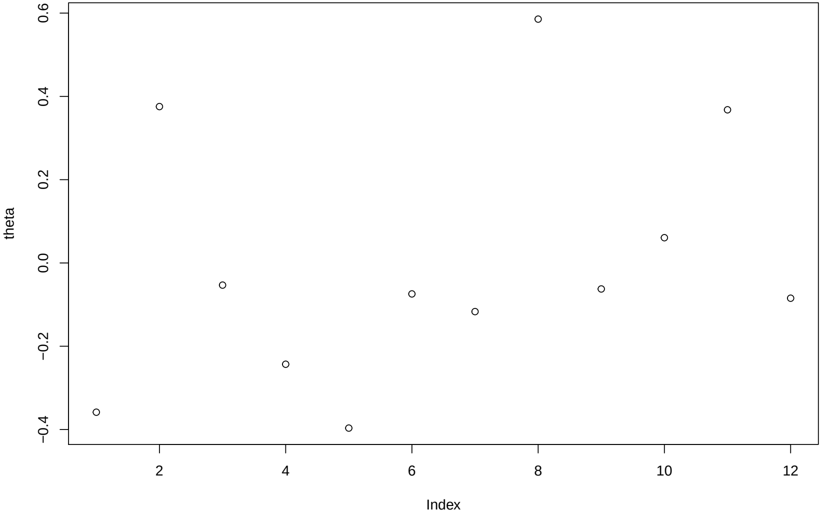
```
## Call:
## bayesx(formula = y ~ sx(hospital, bs = "re"), data = Su
##     family = "binomial", method = "REML")
##
## Fixed effects estimation results:
##
## Parametric coefficients:
```

Anwendung der Approximation III

```
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  -2.5285     0.1378 -18.344 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.
##
## Random effects variances:
##                   Variance Smooth Par.     df Stopped
## sx(hospital):re   0.1444      6.9238 6.9300       0
##
## N = 2814  df = 7.93004  AIC = 1468.71  BIC = 1515.83
## logLik = -726.425  GCV = 0.519216  method = REML  famil
```

```
plot(b$effects$`sx(hospital):re`$Estimate,ylab="theta")
```

Anwendung der Approximation IV



- Bei **Integrated Nested Laplace Approximation**

Anwendung der Approximation V

$$\tilde{p}(\tau|y) = \left. \frac{p(\theta,\tau|y)}{\tilde{p}(\theta|\tau,y)} \right|_{\theta=\theta^*(\tau)}$$

$$\tilde{p}(\theta|y) = \sum_{\tau} \tilde{p}(\theta|\tau,y)\tilde{p}(\tau|y)$$

INLA-Paket kennt drei verschiedene Approximationen (Gauss, Laplace, vereinfachter Laplace), die z.T. auch Schiefe berücksichtigen können.

```
formula = r ~ f(hospital, model="iid",
                param=c(0.001,0.001))
mod.surg = inla(formula, data=Surg,
                family="binomial", Ntrials=n)
```

Anwendung der Approximation VI

```r
plot(mod.surg)
plot(mod.surg$summary.random$hospital$mean)
```

```r
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
```

Anwendung der Approximation VII

```
stanmodel = "
data {
    int<lower=0> n;
    int<lower=0> H;
    int<lower=0> hospital[n];
    int<lower=0> y[n];
    real<lower=0> a;
    real<lower=0> b;
}
parameters {
    vector[H] theta;
    real mu;
    real<lower=0> tau;
}
model {
    for (i in 1:n)
      y[i] ~ bernoulli_logit(theta[hospital[i]]);
    for (i in 1:H)
      theta[i] ~ normal(mu, sqrt(1/tau));
```

Vergleich der Methoden

Besprochene Methoden I

- MCMC selbst implementieren
- Offensichtlich aufwendig
- Kleine Tricks (z.B. log-Akzeptanzwahrscheinlichkeit)
- Besser in z.B. C, C++
- Vorteil: Kann beliebig verändert werden
- STAN / HMC
- (Scheint ziemlich) universell einsetzbar
- Genauer Samplingalgorithmus schwer nachzuvollziehen (Fehler?)
- Eventuell kryptische Fehlermeldungen
- Je nach Modell sehr langsam oder sehr schnell
- BayesX / REML und MCMC
- Nur für bestimmte Modelle (Stukturiert additive Regression, STAR)

Besprochene Methoden II

- MCMC und REML, letzteres sehr schnell
- R-Anbindung (noch) nicht gut, Standalone Software benutzen
- INLA
- Nur für STAR-Modelle, darin aber sehr flexibel
- Kein MCMC, aber i.d.R. sehr gute Approximation
- Dokumentation in R z.T. noch nicht fertig

Alterativen

- OpenBUGS (WinBUGS)
- Klassiker (WinBUGS)
- Generelles MCMC Sampling ähnlich STAN
- Verschiedene, nicht ganz ausgereifte R-Anbindungen, dazu coda
- In der WinBUGS-Variante sehr gute Visualisierung
- JAGS: Neu geschriebene Alternative
- LaplacesDemon (R)
- MCMCpack (R)
- mcmc (R)
- ... (siehe Bayesian Task View in R)
- PyMC (Python)
- emcee (Python)
- Mamba (julia)