Signal Tandmobiel® Data
Time to caries analysis using `bayessurvreg3`

Arnošt Komárek

June 29, 2005

This document describes the analysis of the Signal Tandmobiel® data presented in

Komárek, A. and Lesaffre, E.
Bayesian accelerated failure time model with multivariate doubly-interval-censored data and flexible distributional assumptions.

This article will be refered as Komárek and Lesaffre (2006) and can be found in the `doc` directory of the package `bayesSurv` as `KomarekLesaffre2006.pdf`. For the theory I refer therein.

All R commands presented in this document are available in the same directory as `tandmobCS.R`.

This document should primarily serves as the source of the examples of usage of the functions

- `bayessurvreg3`
- `predictive2`
- `bayesGspline`

Please, take also a look at the extensive help pages of these functions!

0.1 Model

The model considered in this document is the following:

$$\begin{aligned} \log(U_{i,l} - 5.0) &= d_i + \boldsymbol{\delta}' \boldsymbol{z}_{i,l} + \zeta_{i,l}, \qquad i = 1, \ldots, N, \quad l = 1, \ldots, n_i, \qquad (1) \\ \log(V_{i,l} - U_{i,l}) = \log(T_{i,l}) &= b_i + \boldsymbol{\beta}' \boldsymbol{x}_{i,l} + \varepsilon_{i,l}, \qquad i = 1, \ldots, N, \quad l = 1, \ldots, n_i, \qquad (2) \end{aligned}$$

where $U_{i,l}$ is the emergence age of the lth tooth and $V_{i,l}$ is the age when the lth tooth was attacked by caries for the first time. For each child we consider the four permanent first molars (teeth 16, 26, 36, 46), i.e. $n_i \leq 4$. Note that not al teeth are available for all children.

In the initial model (which is not shown in this document but it is discussed by Komárek and Lesaffre, 2006), the covariate vectors were composed of

- $\boldsymbol{z}_{i,l}$ ... Tooth, Girl, Girl:Tooth;
- $\boldsymbol{x}_{i,l}$ ... Tooth, Girl, Brush, Plaque.1, Plaque.2, Seal, Prim5d, Prim5m, Prim5f, Girl:Tooth, Brush:Tooth, Plaque.1:Tooth, Plaque.2:Tooth, Seal:Tooth, Prim5d:Tooth, Prim5m:Tooth, Prim5f:Tooth.

In the final model, the following covariates were used:

- $\boldsymbol{z}_{i,l}$ ... Tooth, Girl;
- $\boldsymbol{x}_{i,l}$ ... Tooth, Girl, Brush, Plaque, Seal, Prim5;

See below for the explanation of the covariates.

1 Initial operations

• Set the directories.

```
> anadir <- "/home/komari/win/work/papers/gsplineTand/anaDMFsymm/"
> chaindir.CS <- paste(anadir, "chains/modelCS", sep = "")
> initdir.CS <- paste(anadir, "chains/end_CS", sep = "")
> plotdir.CS <- paste(anadir, "summPlots/modelCS/", sep = "")
> resultdir <- paste(anadir, "results/", sep = "")
> predCurvesdir <- paste(resultdir, "predCurves/", sep = "")
> figuredir <- "/home/komari/win/work/papers/gsplineTand/RforCRAN/figuresCS/"
```

• We load the package and the data.

```
> library(bayesSurv)
> data(tandmobRoos)
```

In the following we create the dataset with one row per tooth.

• Firstly, create separate datasets for each tooth from the original data. Remove the teeth where there is some missing covariate information.

• The following covariates will be kept.

- `Idnr`: identification number of a child;
- `Maxilla`: binary covariate, 1 = upper (maxillary) tooth, 0 = lower (mandibular) tooth;
- `Tooth`: factor with four levels: 16, 26, 36, 46;
- `Girl`: binary covariate, 1 = girl, 0 = boy;
- `Gender`: factor, girl/boy;
- `Brush`: binary covariate, 1 = brushing at least once a day, 0 = brushing less than once a day;
- `fBrush`: factor, daily = brushing at least once a day, not.daily = brushing less than once a day;
- `Plaque`: binary covariate, 1 = plaque either in pits and fissures or on total occlusal surface, 0 = no plaque;
- `Plaque.1`: dummy, 1 = plaque in pits and fissures, 0 = either no plaque or plaque on total occlusal surface;
- `Plaque.2`: dummy, 1 = plaque on total occlusal surface, 0 = either no plaque or plaque in pits and fissures;
- `fPlaque`: ordered factor, none = no plaque, pits.fiss = plaque in pits and fissures, total = plaque on total occlusal surface;
- `Seal`: binary covariate, 1 = pits and fissures sealing, 0 = no sealing;
- `fSeal`: factor, yes = pits and fissures sealing, no = no sealing;
- `Prim5`: binary covariate concerning the status of the adjacent primary second molar (tooth 55 is adjacent to 16, tooth 65 is adjacent to 26, tooth 75 is adjacent to 36, tooth 85 is adjacent to 46), 1 = either **d**ecayed or **m**issing due to carries or **f**illed, 0 = sound;
- `Prim5d`: dummy for the status of the adjacent primary second molar, 1 = **d**ecayed;
- `Prim5m`: dummy for the status of the adjacent primary second molar, 1 = **m**issing due to caries;
- `Prim5f`: dummy for the status of the adjacent primary second molar, 1 = **f**illed;
- `fPrim5`: factor for the status of the adjacent primary second molar, sound/decayed/filled/ missing.

- `Prim4d`: dummy for the status of the adjacent primary first molar, 1 = **d**ecayed;
- `Prim4m`: dummy for the status of the adjacent primary first molar, 1 = **m**issing due to caries;
- `Prim4f`: dummy for the status of the adjacent primary first molar, 1 = **f**illed;
- `fPrim4`: factor for the status of the adjacent primary first molar, sound/decayed/filled/ missing.

```
> teeth <- c(16, 26, 36, 46)
> maxil <- c(1, 1, 0, 0)
> dmfs <- list()
> for (ti in 1:length(teeth)) {
+     tt <- teeth[ti]
+     remove <- is.na(tandmobRoos[, paste("FBEG.", tt, sep = "")]) |
+         (tandmobRoos[, paste("TOOTH.", tt, sep = "")] == 0) |
+         is.na(tandmobRoos[, paste("T", tt + 39, "d", sep = "")]) |
+         is.na(tandmobRoos[, paste("T", tt + 39, "m", sep = "")]) |
+         is.na(tandmobRoos[, paste("T", tt + 39, "f", sep = "")]) |
+         is.na(tandmobRoos[, paste("T", tt + 39, "s", sep = "")]) |
+         is.na(tandmobRoos[, paste("SEAL.", tt, sep = "")]) |
+         is.na(tandmobRoos[, "FREQ.BR"]) | is.na(tandmobRoos[,
+         paste("PLAQUE.", tt, ".1", sep = "")]) | is.na(tandmobRoos[,
+         paste("PLAQUE.", tt, ".2", sep = "")])
+     temp <- tandmobRoos[!remove, ]
+     temp <- data.frame(Idnr = temp$IDNR, Tooth = rep(tt, dim(temp)[1]),
+         Ebeg = temp[, paste("EBEG.", tt, sep = "")], Eend = temp[,
+             paste("EEND.", tt, sep = "")], Fbeg = temp[, paste("FBEG.",
+             tt, sep = "")], Fend = temp[, paste("FEND.", tt,
+             sep = "")], Maxilla = rep(maxil[ti], dim(temp)[1]),
+         Gender = temp[, "GENDER"], Girl = temp[, "GIRL"], Brush = temp[,
+             "FREQ.BR"], Plaque.1 = temp[, paste("PLAQUE.", tt,
+             ".1", sep = "")], Plaque.2 = temp[, paste("PLAQUE.",
+             tt, ".2", sep = "")], Seal = temp[, paste("SEAL.",
+             tt, sep = "")], Prim5d = temp[, paste("T", tt + 39,
+             "d", sep = "")], Prim5m = temp[, paste("T", tt +
+             39, "m", sep = "")], Prim5f = temp[, paste("T", tt +
+             39, "f", sep = "")], Prim4d = temp[, paste("T", tt +
+             38, "d", sep = "")], Prim4m = temp[, paste("T", tt +
+             38, "m", sep = "")], Prim4f = temp[, paste("T", tt +
+             38, "f", sep = "")])
+     temp$fBrush <- factor(temp$Brush, levels = 0:1, labels = c("not.daily",
+         "daily"))
+     temp$fPlaque <- ordered(0 * (temp$Plaque.1 == 0 & temp$Plaque.2 ==
+         0) + 1 * (temp$Plaque.1 == 1) + 2 * (temp$Plaque.2 ==
+         1), levels = 0:2, labels = c("none", "pits.fiss", "total"))
+     temp$fSeal <- factor(temp$Seal, levels = 0:1, labels = c("no",
+         "yes"))
+     temp$fPrim5 <- factor(0 * (temp$Prim5d == 0 & temp$Prim5m ==
+         0 & temp$Prim5f == 0) + 1 * (temp$Prim5d == 1) + 2 *
+         (temp$Prim5f == 1) + 3 * (temp$Prim5m == 1), levels = 0:3,
+         labels = c("sound", "decayed", "filled", "missing"))
+     temp$fPrim4 <- factor(0 * (temp$Prim4d == 0 & temp$Prim4m ==
+         0 & temp$Prim4f == 0) + 1 * (temp$Prim4d == 1) + 2 *
+         (temp$Prim4m == 1) + 3 * (temp$Prim4f == 1), levels = 0:3,
+         labels = c("sound", "decayed", "missing", "filled"))
+     temp$Plaque <- 0 * (temp$fPlaque == "none") + 1 * (temp$fPlaque ==
+         "pits.fiss" | temp$fPlaque == "total")
+     temp$Prim5 <- 0 * (temp$fPrim5 == "sound") + 1 * (temp$fPrim5 ==
+         "decayed" | temp$fPrim5 == "filled" | temp$fPrim5 ==
+         "missing")
```

```
+     dmfs[[ti]] <- temp
+     rm(list = "temp")
+ }
> names(dmfs) <- teeth
> n.sample <- sapply(dmfs, nrow)
> print(n.sample)

  16   26   36   46
3163 3136 3119 3067
```

• Create a dataset with one row per tooth. Subtract also 5 from each time variable (time zero in our analysis was 5 years of age).

```
> time.0 <- 5
```

```
> data.CS <- rbind(dmfs$"16", dmfs$"26", dmfs$"36", dmfs$"46")
> data.CS <- data.CS[order(data.CS$Idnr), ]
> data.CS$Tooth <- factor(data.CS$Tooth)
> rownames(data.CS) <- 1:dim(data.CS)[1]
> data.CS$Ebeg <- data.CS$Ebeg - time.0
> data.CS$Ebeg[data.CS$Ebeg <= 0] <- NA
> data.CS$Eend <- data.CS$Eend - time.0
> data.CS$Fbeg <- data.CS$Fbeg - time.0
> data.CS$Fbeg[data.CS$Fbeg <= 0] <- NA
> data.CS$Fend <- data.CS$Fend - time.0
> rm(list = c("remove", "ti", "tt", "teeth"))
```

• Show first few rows of the data frame.

```
> print(data.CS[1:10, ])
```

```
   Idnr Tooth Ebeg Eend Fbeg Fend Maxilla Gender Girl Brush Plaque.1 Plaque.2
1     1    16  1.7  2.7  6.2   NA       1   girl    1     0        0        0
2     1    26   NA  2.7  6.2   NA       1   girl    1     0        0        0
3     1    36  1.7  2.7  6.2   NA       0   girl    1     0        0        0
4     1    46  1.7  2.7  6.2   NA       0   girl    1     0        0        0
5     2    16   NA  2.4  2.4  3.4       1    boy    0     0        0        0
6     2    26   NA  2.4  3.4  4.3       1    boy    0     0        1        0
7     2    36   NA  2.4  2.4  3.4       0    boy    0     0        0        0
8     2    46   NA  2.4  2.4  3.4       0    boy    0     0        0        0
9     3    16  0.5  1.5  1.5  2.7       1   girl    1     0        0        0
10    3    26  0.5  1.5  1.5  2.7       1   girl    1     0        1        0
   Seal Prim5d Prim5m Prim5f Prim4d Prim4m Prim4f    fBrush   fPlaque fSeal
1     1      0      0      0      1      0      0 not.daily      none   yes
2     1      1      0      0      1      0      0 not.daily      none   yes
3     1      0      0      0      0      0      1 not.daily      none   yes
4     1      0      0      0      0      0      0 not.daily      none   yes
5     0      0      0      0      0      0      0 not.daily      none    no
6     0      1      0      0      1      0      0 not.daily pits.fiss    no
7     0      1      0      0      0      0      0 not.daily      none    no
8     0      1      0      0      0      1      0 not.daily      none    no
9     0      0      0      0      0      0      0 not.daily      none    no
10    0      0      0      0      0      0      0 not.daily pits.fiss    no
    fPrim5  fPrim4 Plaque Prim5
1    sound decayed      0     0
2  decayed decayed      0     1
3    sound  filled      0     0
4    sound   sound      0     0
```

```
5    sound   sound      0     0
6  decayed decayed      1     1
7  decayed   sound      0     1
8  decayed missing      0     1
9    sound   sound      0     0
10   sound   sound      1     0
```

2 Initial estimates

In this section we compute initial estimates using the maximum-likelihood fits with correctly interval-censored emergence times and the mid-point imputed caries times. A parametric log-logistic AFT model will be used. We will then use these estimates to start the MCMC.

• Initial fit for the emergence.

```
> emerg.CS.aft <- survreg(Surv(Ebeg, Eend, type = "interval2") ~
+     Tooth + Girl, dist = "loglogistic", data = data.CS)
```

• Initial fit for caries.

```
> left <- is.na(data.CS$Ebeg)
> interv <- !is.na(data.CS$Ebeg) & !is.na(data.CS$Eend)
> right <- is.na(data.CS$Eend)
> onset <- data.CS$Ebeg
> onset[left] <- 0.5 * (0 + data.CS$Eend[left])
> onset[right] <- data.CS$Ebeg[right]
> onset[interv] <- 0.5 * (data.CS$Ebeg[interv] + data.CS$Eend[interv])
> vfbeg2 <- data.CS$Fbeg - onset
> vfbeg2[vfbeg2 < 0] <- NA
> vfend2 <- data.CS$Fend - onset
> caries.CS.aft <- survreg(Surv(vfbeg2, vfend2, type = "interval2") ~
+     Tooth + Girl + Brush + Plaque + Seal + Prim5, dist = "loglogistic",
+     data = data.CS)
```

• Results of the initial emergence fits.

```
> summary(emerg.CS.aft)

Call:
survreg(formula = Surv(Ebeg, Eend, type = "interval2") ~ Tooth +
    Girl, data = data.CS, dist = "loglogistic")
              Value Std. Error       z       p
(Intercept)  0.4041    0.00649   62.27 0.00000
Tooth26     -0.0131    0.00833   -1.57 0.11682
Tooth36      0.0132    0.00800    1.66 0.09776
Tooth46      0.0171    0.00804    2.13 0.03356
Girl        -0.0168    0.00571   -2.94 0.00331
Log(scale)  -2.2665    0.01462 -155.05 0.00000

Scale= 0.104

Log logistic distribution
Loglik(model)= -4858.9   Loglik(intercept only)= -4871.6
        Chisq= 25.4 on 4 degrees of freedom, p= 4.2e-05
Number of Newton-Raphson Iterations: 5
n= 12485
```

• Results of the initial caries fits.

```
> summary(caries.CS.aft)

Call:
survreg(formula = Surv(vfbeg2, vfend2, type = "interval2") ~
    Tooth + Girl + Brush + Plaque + Seal + Prim5, data = data.CS,
    dist = "loglogistic")
```

```
               Value Std. Error        z         p
(Intercept)  2.15915     0.0415  52.0543  0.00e+00
Tooth26      0.02916     0.0342   0.8523  3.94e-01
Tooth36     -0.00217     0.0341  -0.0638  9.49e-01
Tooth46     -0.00139     0.0342  -0.0405  9.68e-01
Girl        -0.06496     0.0243  -2.6725  7.53e-03
Brush        0.34537     0.0321  10.7715  4.70e-27
Plaque      -0.24357     0.0260  -9.3787  6.68e-21
Seal         0.07099     0.0278   2.5513  1.07e-02
Prim5       -0.63025     0.0263 -23.9211 1.85e-126
Log(scale)  -0.59707     0.0168 -35.4471 3.21e-275

Scale= 0.55

Log logistic distribution
Loglik(model)= -10585.3   Loglik(intercept only)= -11023
        Chisq= 875.31 on 8 degrees of freedom, p= 0
Number of Newton-Raphson Iterations: 4
n= 12485
```

3 Prior distributions

In this section we specify the prior distributions.

3.1 Priors for the smoothed error distributions

In our model, we have four penalized Gaussian mixtures (G-splines) included: G-spline g_d defining the distribution of the random effects in the emergence part of the model, G-spline $g_\zeta(\zeta)$ defining the distribution of the error term in the emergence part of the model, G-spline g_b defining the distribution of the random effects in the caries part of the model, G-spline $g_\varepsilon(\varepsilon)$ defining the distribution of the error term in the caries part of the model. Since we will not use any informative priors, the prior specifications for the fourG-splines will be the same, namely

- $K = 15$;
- The middle knot is really in the middle, i.e. it is μ_0;
- We use the univariate conditional autoregression prior (`uniCAR`) for the transformed mixture weights $\boldsymbol{a}$ with the differences of the 3rd order;
- The distance between the two knots in each margin is equal to 1.5 times the basis standard deviation σ_1 and σ_2, respectively, i.e. the basis standard deviation is equal to 2/3 times the distance between the two knots;
- Prior distribution for the smoothing hyperparameter λ is Gamma(1, 0.005);
- Prior distribution for the intercept term α is $\mathcal{N}(0, 100)$;
- Prior distribution for the scale parameter τ is Gamma(1, 0.005) on τ^{-2};
- Position of the middle knot γ is fixed (this corresponds to `specification = 2`);
- The basis standard deviation σ is fixed (this corresponds to `specification = 2`).

```
> prior.CS.gspl.emerg <- list(specification = 2, K = 15, izero = 0,
+     neighbor.system = "uniCAR", order = 3, equal.lambda = TRUE,
+     c4delta = 1.5, prior.lambda = "gamma", prior.intercept = "normal",
+     prior.scale = "gamma", prior.gamma = "fixed", prior.sigma = "fixed",
+     shape.lambda = 1, rate.lambda = 0.005, mean.intercept = 0,
+     var.intercept = 100, shape.scale = 1, rate.scale = 0.005)
> prior.CS.gspl.caries <- prior.CS.gspl.emerg
> prior.CS.b.emerg <- prior.CS.gspl.emerg
> prior.CS.b.caries <- prior.CS.gspl.caries
```

3.2 Priors for the regression parameters

For all regression parameters we use $\mathcal{N}(0, 100)$ priors.

```
> prior.CS.beta.emerg <- list(mean.prior = rep(0, 4), var.prior = rep(100,
+     4))
> prior.CS.beta.caries <- list(mean.prior = rep(0, 8), var.prior = rep(100,
+     8))
```

4 Initial values

In this section we specify the initial values for the MCMC sampling. Partially, we use the values obtained in Section 2.

4.1 Emergence part of the model

The following initial values are used:

- $\lambda^{\zeta} = 3\,000$;
- $\alpha^{\zeta} = 0.40$;
- $\tau^{\zeta} = 0.10$;
- $\gamma^{\zeta} = 0$, i.e. the middle knot μ_0^{ζ} is equal to 0. Note that this value remains the same during the whole MCMC;
- $\sigma^{\zeta} = 0.20$ (the basis standard deviations). This value remains the same during the whole MCMC as well;
- $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_4)' = (-0.01,\ 0.01,\ 0.02,\ -0.02)'$;
- $\lambda^{d} = 1\,000$;
- $\alpha^{d} = 0.00$;
- $\tau^{d} = 0.05$;
- $\gamma^{d} = 0$, i.e. the middle knot μ_0^{d} is equal to 0. Note that this value remains the same during the whole MCMC;
- $\sigma^{d} = 0.20$ (the basis standard deviations). This value remains the same during the whole MCMC as well;

```
> init.CS.emerg <- list(lambda = 3000, intercept = 0.4, scale = 0.1,
+     gamma = 0, sigma = 0.2, beta = c(-0.01, 0.01, 0.02, -0.02),
+     lambda.b = 1000, intercept.b = 0, scale.b = 0.05, gamma.b = 0,
+     sigma.b = 0.2)
```

4.2 Caries part of the model

The following initial values are used:

- $\lambda^{\varepsilon} = 3\,000$;
- $\alpha^{\varepsilon} = 2.16$;
- $\tau^{\varepsilon} = 0.55$;
- $\gamma^{\varepsilon} = 0$, i.e. the middle knot μ_0^{ε} is equal to 0. Note that this value remains the same during the whole MCMC;
- $\sigma^{\varepsilon} = 0.20$ (the basis standard deviations). This value remains the same during the whole MCMC as well;
- $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_8)' = (0.03,\ 0.00,\ 0.00,\ -0.06,\ 0.35,\ -0.24,\ 0.07,\ -0.63)'$;
- $\lambda^{b} = 1\,000$;
- $\alpha^{b} = 0.00$;
- $\tau^{b} = 0.05$;

- $\gamma^b = 0$, i.e. the middle knot μ_0^b is equal to 0. Note that this value remains the same during the whole MCMC;
- $\sigma^b = 0.20$ (the basis standard deviations). This value remains the same during the whole MCMC as well;

```
> init.CS.caries <- list(lambda = 3000, intercept = 2.16, scale = 0.55,
+     gamma = 0, sigma = 0.2, beta = c(0.03, 0, 0, -0.06, 0.35,
+         -0.24, 0.07, -0.63), lambda.b = 1000, intercept.b = 0,
+     scale.b = 0.05, gamma.b = 0, sigma.b = 0.2)
```

4.3 Read initial values from files

If we wish to restart the MCMC from some specific point we can read the initial values from files. In the following, the initials are taken from the first row of each file (after skiping possible header). It is assumed that the files are in the directory `initdir.CS`. Note how the function `vecr2matr` is used to transform stored allocation variables.

```
> init.CS.emerg <- list()
> init.CS.emerg$iter <- scan(paste(initdir.CS, "/iteration.sim",
+     sep = ""), skip = 1, nlines = 1)
> init.CS.emerg$lambda <- scan(paste(initdir.CS, "/lambda.sim",
+     sep = ""), skip = 1, nlines = 1)
> init.CS.gspline <- scan(paste(initdir.CS, "/gspline.sim", sep = ""),
+     skip = 1, nlines = 1)
> init.CS.emerg$gamma <- init.CS.gspline[1]
> init.CS.emerg$sigma <- init.CS.gspline[2]
> init.CS.emerg$intercept <- init.CS.gspline[4]
> init.CS.emerg$scale <- init.CS.gspline[5]
> init.CS.emerg$beta <- scan(paste(initdir.CS, "/beta.sim", sep = ""),
+     skip = 1, nlines = 1)
> init.CS.emerg$a <- scan(paste(initdir.CS, "/mlogweight.sim",
+     sep = ""), skip = 1, nlines = 1)
> init.CS.emerg$y <- scan(paste(initdir.CS, "/Y.sim", sep = ""),
+     skip = 0, nlines = 1)
> init.CS.r <- scan(paste(initdir.CS, "/r.sim", sep = ""), skip = 0,
+     nlines = 1)
> init.CS.emerg$r <- vecr2matr(init.CS.r, prior.CS.gspl.emerg$K)
> init.CS.emerg$lambda.b <- scan(paste(initdir.CS, "/lambda_b.sim",
+     sep = ""), skip = 1, nlines = 1)
> init.CS.gspline.b <- scan(paste(initdir.CS, "/gspline_b.sim",
+     sep = ""), skip = 1, nlines = 1)
> init.CS.emerg$gamma.b <- init.CS.gspline.b[1]
> init.CS.emerg$sigma.b <- init.CS.gspline.b[2]
> init.CS.emerg$intercept.b <- init.CS.gspline.b[4]
> init.CS.emerg$scale.b <- init.CS.gspline.b[5]
> init.CS.emerg$a.b <- scan(paste(initdir.CS, "/mlogweight_b.sim",
+     sep = ""), skip = 1, nlines = 1)
> init.CS.emerg$b <- scan(paste(initdir.CS, "/b.sim", sep = ""),
+     skip = 0, nlines = 1)
> init.CS.r.b <- scan(paste(initdir.CS, "/r_b.sim", sep = ""),
+     skip = 0, nlines = 1)
> init.CS.emerg$r.b <- vecr2matr(init.CS.r.b, prior.CS.b.emerg$K)
> init.CS.caries <- list()
> init.CS.caries$iter <- scan(paste(initdir.CS, "/iteration.sim",
+     sep = ""), skip = 1, nlines = 1)
> init.CS.caries$lambda <- scan(paste(initdir.CS, "/lambda_2.sim",
+     sep = ""), skip = 1, nlines = 1)
```

```
> init.CS.gspline <- scan(paste(initdir.CS, "/gspline_2.sim", sep = ""),
+     skip = 1, nlines = 1)
> init.CS.caries$gamma <- init.CS.gspline[1]
> init.CS.caries$sigma <- init.CS.gspline[2]
> init.CS.caries$intercept <- init.CS.gspline[4]
> init.CS.caries$scale <- init.CS.gspline[5]
> init.CS.caries$beta <- scan(paste(initdir.CS, "/beta_2.sim",
+     sep = ""), skip = 1, nlines = 1)
> init.CS.caries$a <- scan(paste(initdir.CS, "/mlogweight_2.sim",
+     sep = ""), skip = 1, nlines = 1)
> init.CS.caries$y <- scan(paste(initdir.CS, "/Y_2.sim", sep = ""),
+     skip = 0, nlines = 1)
> init.CS.r <- scan(paste(initdir.CS, "/r_2.sim", sep = ""), skip = 0,
+     nlines = 1)
> init.CS.caries$r <- vecr2matr(init.CS.r, prior.CS.gspl.caries$K)
> init.CS.caries$lambda.b <- scan(paste(initdir.CS, "/lambda_b2.sim",
+     sep = ""), skip = 1, nlines = 1)
> init.CS.gspline.b <- scan(paste(initdir.CS, "/gspline_b2.sim",
+     sep = ""), skip = 1, nlines = 1)
> init.CS.caries$gamma.b <- init.CS.gspline.b[1]
> init.CS.caries$sigma.b <- init.CS.gspline.b[2]
> init.CS.caries$intercept.b <- init.CS.gspline.b[4]
> init.CS.caries$scale.b <- init.CS.gspline.b[5]
> init.CS.caries$a.b <- scan(paste(initdir.CS, "/mlogweight_b2.sim",
+     sep = ""), skip = 1, nlines = 1)
> init.CS.caries$b <- scan(paste(initdir.CS, "/b_2.sim", sep = ""),
+     skip = 0, nlines = 1)
> init.CS.r.b <- scan(paste(initdir.CS, "/r_b2.sim", sep = ""),
+     skip = 0, nlines = 1)
> init.CS.caries$r.b <- vecr2matr(init.CS.r.b, prior.CS.b.caries$K)
```

• Show the initial values for the emergence process using the standard function `str`.

```
> str(init.CS.emerg)
```

```
List of 18
 $ iter       : num 1100000
 $ lambda     : num 0.0303
 $ gamma      : num 0
 $ sigma      : num 0.2
 $ intercept  : num 0.599
 $ scale      : num 0.124
 $ beta       : num [1:4] -0.00111  0.00987  0.00100 -0.01245
 $ a          : num [1:31] -55.9 -40.0 -29.8 -23.2 -21.9 ...
 $ y          : num [1:12485] 0.631 0.624 0.625 0.631 0.695 ...
 $ r          : num [1:12485] 3 3 3 3 3 3 3 3 3 3 ...
 $ lambda.b   : num 507
 $ gamma.b    : num 0
 $ sigma.b    : num 0.2
 $ intercept.b: num 0
 $ scale.b    : num 0.134
 $ a.b        : num [1:31] 8.63 8.47 8.36 8.32 8.31 ...
 $ b          : num [1:3520] -0.0685 -0.0194 -0.5945 -0.1389 -0.3916 ...
 $ r.b        : num [1:3520] -2 0 -15 -3 -9 0 -15 0 -1 -11 ...
```

• Show the initial values for the caries process using the standard function `str`.

```
> str(init.CS.caries)
```

```
List of 18
 $ iter       : num 1100000
 $ lambda     : num 0.121
 $ gamma      : num 0
 $ sigma      : num 0.2
 $ intercept  : num 0.84
 $ scale      : num 0.47
 $ beta       : num [1:8] -0.00839  0.01603 -0.01372 -0.10529  0.35517 ...
 $ a          : num [1:31] -27.67 -13.44  -5.74  -3.69  -3.93 ...
 $ y          : num [1:12485] 2.183 3.470 2.060 3.000 0.221 ...
 $ r          : num [1:12485] 1 11 1 7 1 6 1 1 -11 -7 ...
 $ lambda.b   : num 835
 $ gamma.b    : num 0
 $ sigma.b    : num 0.2
 $ intercept.b: num 0
 $ scale.b    : num 0.279
 $ a.b        : num [1:31] 7.27 6.92 6.66 6.51 6.47 ...
 $ b          : num [1:3520]  1.114 -0.816  0.291 -0.823  0.828 ...
 $ r.b        : num [1:3520] 14 -9 4 -9 10 10 -2 -13 2 10 ...
```

5 MCMC sampling – function `bayessurvreg3`

In this section we start the MCMC sampling. The chains are stored in the directory `chaindir.CS`.

• Specify the length of the simulation.

```
> nsimul.CS <- list(niter = 1100000, nthin = 3, nburn = 1e+06,
+     nwrite = 250)
```

• Sample.

```
> sample.CS <- bayessurvreg3(formula = Surv(Ebeg, Eend, type = "interval2") ~
+     Tooth + Girl + cluster(Idnr), random = ~1, formula2 = Surv(Fbeg,
+     Fend, type = "interval2") ~ Tooth + Girl + Brush + Plaque +
+     Seal + Prim5 + cluster(Idnr), random2 = ~1, onlyX = FALSE,
+     dir = chaindir.CS, nsimul = nsimul.CS, prior = prior.CS.gspl.emerg,
+     prior2 = prior.CS.gspl.caries, prior.beta = prior.CS.beta.emerg,
+     prior.beta2 = prior.CS.beta.caries, prior.b = prior.CS.b.emerg,
+     prior.b2 = prior.CS.b.caries, init = init.CS.emerg, init2 = init.CS.caries,
+     store = list(a = TRUE, a2 = TRUE, a.b = TRUE, a.b2 = TRUE),
+     data = data.CS)
```

6 Posterior predictive survivor, density and hazard – function predictive2

Once the MCMC sampling is finished we can compute predictive survivor density and hazard functions for some specific combinations of covariates. In this section we show how the predictive quantities can be computed for the caries part of the model.

• Specify whether some sampled values should be skipped, how often will the information concerning the progress of computation be print on the screen. Further specify the grid of values in which the predictive quantities will be computed, specify probabilities of quantiles we want to compute.

```
> skip <- 0
> nwrite <- 5000
> pred.grid <- c(seq(0.1, 1.5, by = 0.05), seq(1.65, 6, by = 0.15))
> quants <- c(0.025, 0.5, 0.975)
> nquant <- length(quants)
```

6.1 Combinations of covariates

• Specify the combination of covariates for which the predictive quantities for each tooth will be computed.

```
> pGender <- factor(c("boy", "girl"))
> pBrush <- factor(c("not.daily", "daily"))
> pSeal <- factor(c("no", "yes"))
> pPlaque <- factor(c("none", "present"))
> pPrim5 <- factor(c("sound", "dmf"))
> pdata <- expand.grid(ffPrim5 = pPrim5, ffPlaque = pPlaque, fSeal = pSeal,
+     fBrush = pBrush, Gender = pGender)
> pdata$Prim5 <- 1 * (pdata$ffPrim5 == "dmf")
> pdata$Plaque <- 1 * (pdata$ffPlaque == "present")
> pdata$Seal <- 1 * (pdata$fSeal == "yes")
> pdata$Brush <- 1 * (pdata$fBrush == "daily")
> pdata$Girl <- 1 * (pdata$Gender == "girl")
> ncov <- nrow(pdata)
> pred.data <- data.frame(Idnr = 1:(4 * ncov), Tooth = factor(c(rep(16,
+     ncov), rep(26, ncov), rep(36, ncov), rep(46, ncov))))
> pred.data$Tooth26 <- 1 * (pred.data$Tooth == 26)
> pred.data$Tooth36 <- 1 * (pred.data$Tooth == 36)
> pred.data$Tooth46 <- 1 * (pred.data$Tooth == 46)
> pred.data <- cbind(pred.data, rbind(pdata, pdata, pdata, pdata))
```

• Look at the data frame we use for the prediction. Here I print only first 32 rows, which is repeated 4 times with the only difference for values of the covariates Tooth26, Tooth36, Tooth46.

```
> print(pred.data[1:ncov, ])
```

```
  Idnr Tooth Tooth26 Tooth36 Tooth46 ffPrim5 ffPlaque fSeal    fBrush Gender
1    1    16       0       0       0   sound     none    no not.daily    boy
2    2    16       0       0       0     dmf     none    no not.daily    boy
3    3    16       0       0       0   sound  present    no not.daily    boy
4    4    16       0       0       0     dmf  present    no not.daily    boy
5    5    16       0       0       0   sound     none   yes not.daily    boy
6    6    16       0       0       0     dmf     none   yes not.daily    boy
7    7    16       0       0       0   sound  present   yes not.daily    boy
8    8    16       0       0       0     dmf  present   yes not.daily    boy
9    9    16       0       0       0   sound     none    no     daily    boy
```

```
10   10    16        0        0        0     dmf     none    no      daily    boy
11   11    16        0        0        0   sound  present    no      daily    boy
12   12    16        0        0        0     dmf  present    no      daily    boy
13   13    16        0        0        0   sound     none   yes      daily    boy
14   14    16        0        0        0     dmf     none   yes      daily    boy
15   15    16        0        0        0   sound  present   yes      daily    boy
16   16    16        0        0        0     dmf  present   yes      daily    boy
17   17    16        0        0        0   sound     none    no not.daily   girl
18   18    16        0        0        0     dmf     none    no not.daily   girl
19   19    16        0        0        0   sound  present    no not.daily   girl
20   20    16        0        0        0     dmf  present    no not.daily   girl
21   21    16        0        0        0   sound     none   yes not.daily   girl
22   22    16        0        0        0     dmf     none   yes not.daily   girl
23   23    16        0        0        0   sound  present   yes not.daily   girl
24   24    16        0        0        0     dmf  present   yes not.daily   girl
25   25    16        0        0        0   sound     none    no      daily   girl
26   26    16        0        0        0     dmf     none    no      daily   girl
27   27    16        0        0        0   sound  present    no      daily   girl
28   28    16        0        0        0     dmf  present    no      daily   girl
29   29    16        0        0        0   sound     none   yes      daily   girl
30   30    16        0        0        0     dmf     none   yes      daily   girl
31   31    16        0        0        0   sound  present   yes      daily   girl
32   32    16        0        0        0     dmf  present   yes      daily   girl
   Prim5 Plaque Seal Brush Girl
1      0      0    0     0    0
2      1      0    0     0    0
3      0      1    0     0    0
4      1      1    0     0    0
5      0      0    1     0    0
6      1      0    1     0    0
7      0      1    1     0    0
8      1      1    1     0    0
9      0      0    0     1    0
10     1      0    0     1    0
11     0      1    0     1    0
12     1      1    0     1    0
13     0      0    1     1    0
14     1      0    1     1    0
15     0      1    1     1    0
16     1      1    1     1    0
17     0      0    0     0    1
18     1      0    0     0    1
19     0      1    0     0    1
20     1      1    0     0    1
21     0      0    1     0    1
22     1      0    1     0    1
23     0      1    1     0    1
24     1      1    1     0    1
25     0      0    0     1    1
26     1      0    0     1    1
27     0      1    0     1    1
28     1      1    0     1    1
29     0      0    1     1    1
30     1      0    1     1    1
31     0      1    1     1    1
32     1      1    1     1    1
```

6.2 Compute predictive quantities

• To avoid problems with memory (when the quantiles are asked, all predictive quantities during all the MCMC iterations must be at some moment stored in the memory which might often kill R due to memory problems), we compute predictive functions always only for two combinations of covariates at once and then put the results together.

• Here we define the combinations of covariates for which the predictive curves will be computed at once.

```
> tooth <- c(16, 26, 36, 46)
> start <- list(tooth16 = seq(1, ncov, by = 2), tooth26 = seq(ncov +
+     1, 2 * ncov, by = 2), tooth36 = seq(2 * ncov + 1, 3 * ncov,
+     by = 2), tooth46 = seq(3 * ncov + 1, 4 * ncov, by = 2))
> end <- list(tooth16 = seq(2, ncov, by = 2), tooth26 = seq(ncov +
+     2, 2 * ncov, by = 2), tooth36 = seq(2 * ncov + 2, 3 * ncov,
+     by = 2), tooth46 = seq(3 * ncov + 2, 4 * ncov, by = 2))
```

• Compute the predictive quantities using the function `predictive2`. The loop over `k` is the loop over the four teeth. The loop over `ii` is the loop over the sets of covariate combinations which are processed at once.

The list `pred.CS` will have four components (one per tooth) and each component will again be the list with each component corresponding to the set of covariate combinations that were processed at once.

In the model formula I use dummies for Tooth created by hand. However the factor variable could also be used.

```
> pred.CS <- list()
> for (k in 1:4) {
+     cat("Performing TOOTH ", tooth[k], "\n", sep = "")
+     pred.CS[[k]] <- list()
+     for (ii in 1:length(start[[k]])) {
+         cat("Performing the covariate set number ", ii, "\n",
+             sep = "")
+         pdata.now <- pred.data[start[[k]][ii]:end[[k]][ii], ]
+         nr <- nrow(pdata.now)
+         pred.CS[[k]][[ii]] <- predictive2(Surv(rep(1, nr), rep(1,
+             nr)) ~ Tooth26 + Tooth36 + Tooth46 + Girl + Brush +
+             Plaque + Seal + Prim5 + cluster(Idnr), random = ~1,
+             grid = pred.grid, data = pdata.now, Gspline = list(dim = 1,
+                 K = 15), quantile = quants, skip = skip, by = 1,
+             nwrite = nwrite, only.aver = FALSE, predict = list(density = TRUE,
+                 Surv = TRUE, hazard = TRUE), dir = chaindir.CS,
+             extens = "_2", extens.random = "_b2", version = 3)
+     }
+ }
> names(pred.CS) <- paste(tooth)
```

• With the following code we create the list `predd.CS` with four components (one per tooth). Each of its components will join the results for all sets of covariates that were separately processed. At the end we assign `predd.CS` to `pred.CS`.

```
> predd.CS <- list()
> for (tt in 1:4) {
+     predd.CS[[tt]] <- list()
+     predd.CS[[tt]]$grid <- pred.CS[[tt]][[1]]$grid
+     predd.CS[[tt]]$Surv <- pred.CS[[tt]][[1]]$Surv
+     predd.CS[[tt]]$hazard <- pred.CS[[tt]][[1]]$hazard
+     predd.CS[[tt]]$density <- pred.CS[[tt]][[1]]$density
```

```
+     predd.CS[[tt]]$quant.Surv <- pred.CS[[tt]][[1]]$quant.Surv
+     predd.CS[[tt]]$quant.hazard <- pred.CS[[tt]][[1]]$quant.hazard
+     predd.CS[[tt]]$quant.density <- pred.CS[[tt]][[1]]$quant.density
+     for (ii in 2:length(start[[tt]])) {
+         predd.CS[[tt]]$Surv <- rbind(predd.CS[[tt]]$Surv, pred.CS[[tt]][[ii]]$Surv)
+         predd.CS[[tt]]$hazard <- rbind(predd.CS[[tt]]$hazard,
+             pred.CS[[tt]][[ii]]$hazard)
+         predd.CS[[tt]]$density <- rbind(predd.CS[[tt]]$density,
+             pred.CS[[tt]][[ii]]$density)
+         templ <- length(predd.CS[[tt]]$quant.Surv)
+         for (ij in 1:length(pred.CS[[tt]][[ii]]$quant.Surv)) {
+             predd.CS[[tt]]$quant.Surv[[templ + ij]] <- pred.CS[[tt]][[ii]]$quant.Surv[[ij]]
+             predd.CS[[tt]]$quant.hazard[[templ + ij]] <- pred.CS[[tt]][[ii]]$quant.hazard[[ij]]
+             predd.CS[[tt]]$quant.density[[templ + ij]] <- pred.CS[[tt]][[ii]]$quant.density[[ij]]
+         }
+     }
+ }
> names(predd.CS) <- paste(tooth)
> pred.CS <- predd.CS
> rm(list = "predd.CS")
```

6.3 Store predictive quantities in files

We can also store predictive quantities in files for future use. We store them in the directory `predCurvesdir`.

• Posterior predictive means. In each file the first row is the grid in which the predictive functions are evaluated, the 2nd to 33rd rows are predictive functions for each combination of covariates specified by `pred.data[1:32,]`. We create one file per tooth.

```
> for (tt in 1:4) {
+     sink(paste(predCurvesdir, "predDensCaries.", tooth[tt], ".CS.",
+         "dat", sep = ""))
+     cat(pred.CS[[tt]]$grid, "\n", sep = "   ")
+     for (j in 1:ncov) cat(pred.CS[[tt]]$density[j, ], "\n", sep = "   ")
+     sink()
+     sink(paste(predCurvesdir, "predSurvCaries.", tooth[tt], ".CS.",
+         "dat", sep = ""))
+     cat(pred.CS[[tt]]$grid, "\n", sep = "   ")
+     for (j in 1:ncov) cat(pred.CS[[tt]]$Surv[j, ], "\n", sep = "   ")
+     sink()
+     sink(paste(predCurvesdir, "predhazardCaries.", tooth[tt],
+         ".CS.", "dat", sep = ""))
+     cat(pred.CS[[tt]]$grid, "\n", sep = "   ")
+     for (j in 1:ncov) cat(pred.CS[[tt]]$hazard[j, ], "\n", sep = "   ")
+     sink()
+ }
```

• Posterior predictive quantiles. In each file the first row is the grid in which the predictive functions are evaluated, the 2nd to 4th row are 2.5%, 50% and 97.5% quantile for the first combination of covariates, the 5th to 7th row are 2.5%, 50% and 97.5% quantile for the second combination of covariates etc. Again, one file per tooth is created.

```
> for (tt in 1:4) {
+     sink(paste(predCurvesdir, "quantDensCaries.", tooth[tt],
+         ".CS.", "dat", sep = ""))
+     cat(pred.CS[[tt]]$grid, "\n", sep = "   ")
+     for (j in 1:ncov) {
+         for (jj in 1:nquant) cat(pred.CS[[tt]]$quant.density[[j]][jj,
```

```
+             ], "\n", sep = "   ")
+     }
+     sink()
+     sink(paste(predCurvesdir, "quantSurvCaries.", tooth[tt],
+         ".CS.", "dat", sep = ""))
+     cat(pred.CS[[tt]]$grid, "\n", sep = "   ")
+     for (j in 1:ncov) {
+         for (jj in 1:nquant) cat(pred.CS[[tt]]$quant.Surv[[j]][jj,
+             ], "\n", sep = "   ")
+     }
+     sink()
+     sink(paste(predCurvesdir, "quanthazardCaries.", tooth[tt],
+         ".CS.", "dat", sep = ""))
+     cat(pred.CS[[tt]]$grid, "\n", sep = "   ")
+     for (j in 1:ncov) {
+         for (jj in 1:nquant) cat(pred.CS[[tt]]$quant.hazard[[j]][jj,
+             ], "\n", sep = "   ")
+     }
+     sink()
+ }
```

6.4 Read the predictive quantities and plot them

We read the predictive quantities from files and plot some of them.

• Some useful variables.

```
> tnames <- c("16", "26", "36", "46")
> qnames <- c("2.5%", "50%", "97.5%")
> nquant <- length(qnames)
> ncov <- 32
```

• Read posterior predictive means.

```
> pred.CS <- list()
> for (tt in 1:4) {
+     pred.CS[[tt]] <- list()
+     pred.CS[[tt]]$grid <- scan(paste(predCurvesdir, "predDensCaries.",
+         tnames[tt], ".CS.dat", sep = ""), nlines = 1)
+     ngrid <- length(pred.CS[[tt]]$grid)
+     pred.CS[[tt]]$density <- matrix(scan(paste(predCurvesdir,
+         "predDensCaries.", tnames[tt], ".CS.dat", sep = ""),
+         skip = 1), ncol = ngrid, byrow = TRUE)
+     pred.CS[[tt]]$Surv <- matrix(scan(paste(predCurvesdir, "predSurvCaries.",
+         tnames[tt], ".CS.dat", sep = ""), skip = 1), ncol = ngrid,
+         byrow = TRUE)
+     pred.CS[[tt]]$hazard <- matrix(scan(paste(predCurvesdir,
+         "predhazardCaries.", tnames[tt], ".CS.dat", sep = ""),
+         skip = 1), ncol = ngrid, byrow = TRUE)
+ }
> names(pred.CS) <- tnames
```

• Read posterior predictive quantiles.

```
> for (tt in 1:4) {
+     ngrid <- length(pred.CS[[tt]]$grid)
+     pred.CS[[tt]]$quant.density <- list()
+     pred.CS[[tt]]$quant.Surv <- list()
```

```
+     pred.CS[[tt]]$quant.hazard <- list()
+     for (j in 1:ncov) {
+         pred.CS[[tt]]$quant.density[[j]] <- matrix(scan(paste(predCurvesdir,
+             "quantDensCaries.", tnames[tt], ".CS.dat", sep = ""),
+             skip = nquant * (j - 1) + 1, nlines = nquant), ncol = ngrid,
+             byrow = TRUE)
+         pred.CS[[tt]]$quant.Surv[[j]] <- matrix(scan(paste(predCurvesdir,
+             "quantSurvCaries.", tnames[tt], ".CS.dat", sep = ""),
+             skip = nquant * (j - 1) + 1, nlines = nquant), ncol = ngrid,
+             byrow = TRUE)
+         pred.CS[[tt]]$quant.hazard[[j]] <- matrix(scan(paste(predCurvesdir,
+             "quanthazardCaries.", tnames[tt], ".CS.dat", sep = ""),
+             skip = nquant * (j - 1) + 1, nlines = nquant), ncol = ngrid,
+             byrow = TRUE)
+         rownames(pred.CS[[tt]]$quant.density[[j]]) <- qnames
+         rownames(pred.CS[[tt]]$quant.Surv[[j]]) <- qnames
+         rownames(pred.CS[[tt]]$quant.hazard[[j]]) <- qnames
+     }
+ }
```

• Some parameters for the plots.

```
> pdfwidth <- 6
> pdfheight <- 9
> teeth <- c(16, 26, 36, 46)
```

• Combinations of covariates we have.

```
> pGender <- factor(c("boy", "girl"))
> pBrush <- factor(c("not.daily", "daily"))
> pSeal <- factor(c("no", "yes"))
> pPlaque <- factor(c("none", "present"))
> pPrim5 <- factor(c("sound", "dmf"))
> pdata <- expand.grid(ffPrim5 = pPrim5, ffPlaque = pPlaque, fSeal = pSeal,
+     fBrush = pBrush, Gender = pGender)
> pdata$Prim5 <- 1 * (pdata$ffPrim5 == "dmf")
> pdata$Plaque <- 1 * (pdata$ffPlaque == "present")
> pdata$Seal <- 1 * (pdata$fSeal == "yes")
> pdata$Brush <- 1 * (pdata$fBrush == "daily")
> pdata$Girl <- 1 * (pdata$Gender == "girl")
> ncov <- nrow(pdata)
> print(pdata)
```

```
   ffPrim5 ffPlaque fSeal    fBrush Gender Prim5 Plaque Seal Brush Girl
1    sound     none    no not.daily    boy     0      0    0     0    0
2      dmf     none    no not.daily    boy     1      0    0     0    0
3    sound  present    no not.daily    boy     0      1    0     0    0
4      dmf  present    no not.daily    boy     1      1    0     0    0
5    sound     none   yes not.daily    boy     0      0    1     0    0
6      dmf     none   yes not.daily    boy     1      0    1     0    0
7    sound  present   yes not.daily    boy     0      1    1     0    0
8      dmf  present   yes not.daily    boy     1      1    1     0    0
9    sound     none    no     daily    boy     0      0    0     1    0
10     dmf     none    no     daily    boy     1      0    0     1    0
11   sound  present    no     daily    boy     0      1    0     1    0
12     dmf  present    no     daily    boy     1      1    0     1    0
13   sound     none   yes     daily    boy     0      0    1     1    0
14     dmf     none   yes     daily    boy     1      0    1     1    0
```

```
15   sound  present   yes     daily    boy     0      1    1     1    0
16     dmf  present   yes     daily    boy     1      1    1     1    0
17   sound     none    no not.daily   girl     0      0    0     0    1
18     dmf     none    no not.daily   girl     1      0    0     0    1
19   sound  present    no not.daily   girl     0      1    0     0    1
20     dmf  present    no not.daily   girl     1      1    0     0    1
21   sound     none   yes not.daily   girl     0      0    1     0    1
22     dmf     none   yes not.daily   girl     1      0    1     0    1
23   sound  present   yes not.daily   girl     0      1    1     0    1
24     dmf  present   yes not.daily   girl     1      1    1     0    1
25   sound     none    no     daily   girl     0      0    0     1    1
26     dmf     none    no     daily   girl     1      0    0     1    1
27   sound  present    no     daily   girl     0      1    0     1    1
28     dmf  present    no     daily   girl     1      1    0     1    1
29   sound     none   yes     daily   girl     0      0    1     1    1
30     dmf     none   yes     daily   girl     1      0    1     1    1
31   sound  present   yes     daily   girl     0      1    1     1    1
32     dmf  present   yes     daily   girl     1      1    1     1    1
```

• Predictive survivor functions, one covariate combination per plot and 95% confidence region. See Figure 1 for the plot of the tooth 16, boys and first 8 covariate combinations.

```
> sets <- list(set1 = 1:8, set2 = 9:16, set3 = 17:24, set4 = 25:32)
> label <- paste(pdata$Gender, ", brush: ", pdata$fBrush, ", seal: ",
+     pdata$fSeal, ", plaq: ", pdata$ffPlaque, ", ", pdata$ffPrim5,
+     sep = "")
> pdf(paste(figuredir, "predSurvCI_CS.pdf", sep = ""), width = pdfwidth,
+     height = pdfheight)
> for (tt in 1:4) {
+     for (ss in 1:length(sets)) {
+         par(mfrow = c(4, 2), bty = "n", lwd = 1.2)
+         for (ii in sets[[ss]]) {
+             plot(pred.CS[[tt]]$grid, pred.CS[[tt]]$Surv[ii, ],
+                 type = "l", lty = 1, xlab = "Time since emergence (years)",
+                 ylab = "Caries free", ylim = c(0, 1))
+             lines(pred.CS[[tt]]$grid, pred.CS[[tt]]$quant.Surv[[ii]]["2.5%",
+                 ], lty = 2)
+             lines(pred.CS[[tt]]$grid, pred.CS[[tt]]$quant.Surv[[ii]]["97.5%",
+                 ], lty = 2)
+             title(main = paste("Tooth ", teeth[tt], sep = ""),
+                 sub = label[ii])
+         }
+     }
+ }
> dev.off()
```

• Predictive hazard functions, one covariate combination per plot and 95% confidence region. See Figure 2 for the plot of the tooth 16, boys and first 8 covariate combinations.

```
> pdf(paste(figuredir, "predhazardCI_CS.pdf", sep = ""), width = pdfwidth,
+     height = pdfheight)
> ylim <- NULL
> for (tt in 1:4) {
+     for (ss in 1:length(sets)) {
+         par(mfrow = c(4, 2), bty = "n", lwd = 1.2)
+         for (ii in sets[[ss]]) {
+             plot(pred.CS[[tt]]$grid, pred.CS[[tt]]$quant.hazard[[ii]]["97.5%",
+                 ], type = "l", lty = 2, xlab = "Time since emergence (years)",
+                 ylab = "Hazard for caries", ylim = ylim)
```

```
+             lines(pred.CS[[tt]]$grid, pred.CS[[tt]]$hazard[ii,
+                 ], lty = 1)
+             lines(pred.CS[[tt]]$grid, pred.CS[[tt]]$quant.hazard[[ii]]["2.5%",
+                 ], lty = 2)
+             title(main = paste("Tooth ", teeth[tt], sep = ""),
+                 sub = label[ii])
+         }
+     }
+ }
> dev.off()
```

• Posterior predictive means of the survivor functions. 16 covariate combinations per plot, 1 plot per tooth and gender. See Figure 3 for the plot of the tooth 16 and boys.

red = *not.daily brushers* and *dmf*;
orange = *not.daily brushers* and *sound*;
blue = *daily brushers* and *dmf*;
darkblue = *daily brushers* and *sound*;

Line type: 2 = *no plaque, no seal*; 4 = *present plaque, no seal*; 1 = *no plaque, sealed*; 3 = *present plaque, sealed*.

```
> col <- c(rep(c("orange", "red"), 4), rep(c("darkblue", "blue"),
+     4))
> lty <- rep(c(2, 2, 4, 4, 1, 1, 3, 3), 2)
> glabel <- c("Boy", "Girl")
> ncov.gender <- ncov/2
> pdf(paste(figuredir, "predSurv_CS.pdf", sep = ""), width = pdfheight,
+     height = pdfwidth)
> for (tt in 1:4) {
+     for (gg in 1:2) {
+         par(mfrow = c(1, 1), bty = "n", lwd = 1.2)
+         plot(pred.CS[[tt]]$grid, pred.CS[[tt]]$Surv[(gg - 1) *
+             ncov.gender + 1, ], type = "l", lty = lty[1], col = col[1],
+             xlab = "Time since emergence (years)", ylab = "Caries free",
+             ylim = c(0, 1))
+         for (i in 2:ncov.gender) {
+             lines(pred.CS[[tt]]$grid, pred.CS[[tt]]$Surv[(gg -
+                 1) * ncov.gender + i, ], lty = lty[i], col = col[i])
+         }
+         title(main = paste("Tooth ", teeth[tt], ", ", glabel[gg],
+             sep = ""))
+         legend(0, 0.7, legend = c("No plaque, sealed", "No plaque, not sealed",
+             "Plaque, sealed", "Plaque, not sealed"), lty = 1:4,
+             bty = "n", y.intersp = 1.2, col = "black")
+         legend(0, 0.3, legend = c("Daily brush, sound primary",
+             "Daily brush, dmf primary", "Not daily brush, sound primary",
+             "Not daily brush, dmf primary"), lty = 1, bty = "n",
+             y.intersp = 1.2, col = c("darkblue", "blue", "orange",
+                 "red"))
+     }
+ }
> dev.off()
```

• Posterior predictive means of the hazard functions. 16 covariate combinations per plot, 1 plot per tooth and gender. See Figure 4 for the plot of the tooth 16 and boys. Format is the same as for the survivor functions.

```
> pdf(paste(figuredir, "predhazard_CS.pdf", sep = ""), width = pdfheight,
```

```
+     height = pdfwidth)
> ylim <- c(0, 0.4)
> for (tt in 1:4) {
+     for (gg in 1:2) {
+         par(mfrow = c(1, 1), bty = "n", lwd = 1.2)
+         plot(pred.CS[[tt]]$grid, pred.CS[[tt]]$hazard[(gg - 1) *
+             ncov.gender + 1, ], type = "l", lty = lty[1], col = col[1],
+             xlab = "Time since emergence (years)", ylab = "Hazard for caries",
+             ylim = ylim)
+         for (i in 2:ncov.gender) {
+             lines(pred.CS[[tt]]$grid, pred.CS[[tt]]$hazard[(gg -
+                 1) * ncov.gender + i, ], lty = lty[i], col = col[i])
+         }
+         title(main = paste("Tooth ", teeth[tt], ", ", glabel[gg],
+             sep = ""))
+         legend(0, 0.4, legend = c("No plaque, sealed", "No plaque, not sealed",
+             "Plaque, sealed", "Plaque, not sealed"), lty = 1:4,
+             bty = "n", y.intersp = 1.2, col = "black")
+         legend(0, 0.3, legend = c("Daily brush, sound primary",
+             "Daily brush, dmf primary", "Not daily brush, sound primary",
+             "Not daily brush, dmf primary"), lty = 1, bty = "n",
+             y.intersp = 1.2, col = c("darkblue", "blue", "orange",
+                 "red"))
+     }
+ }
> dev.off()
```

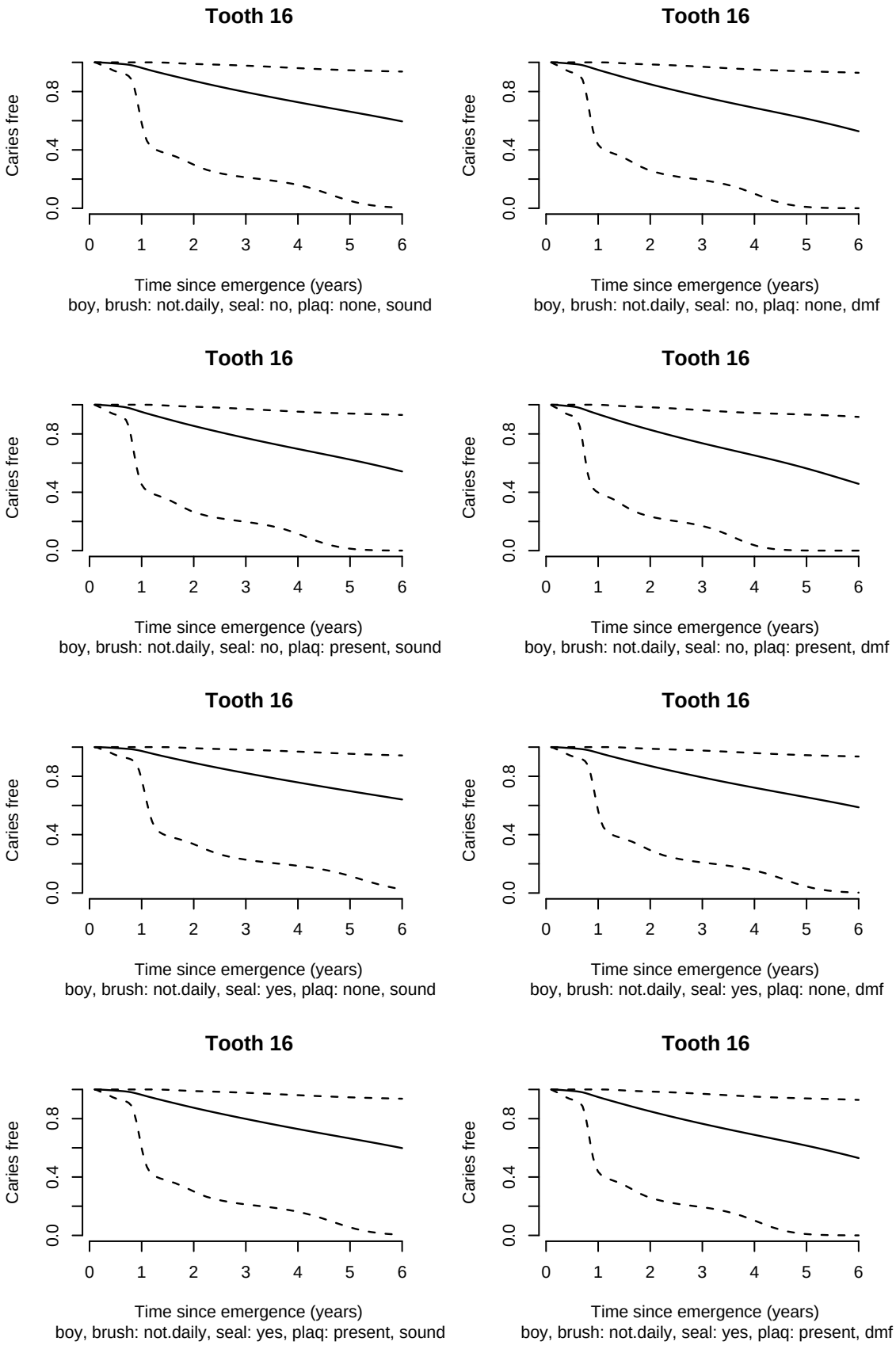


Figure 1: Predictive survivor curves (posterior predictive means and 95% pointwise credible regions) for caries for tooth 16 of boys.

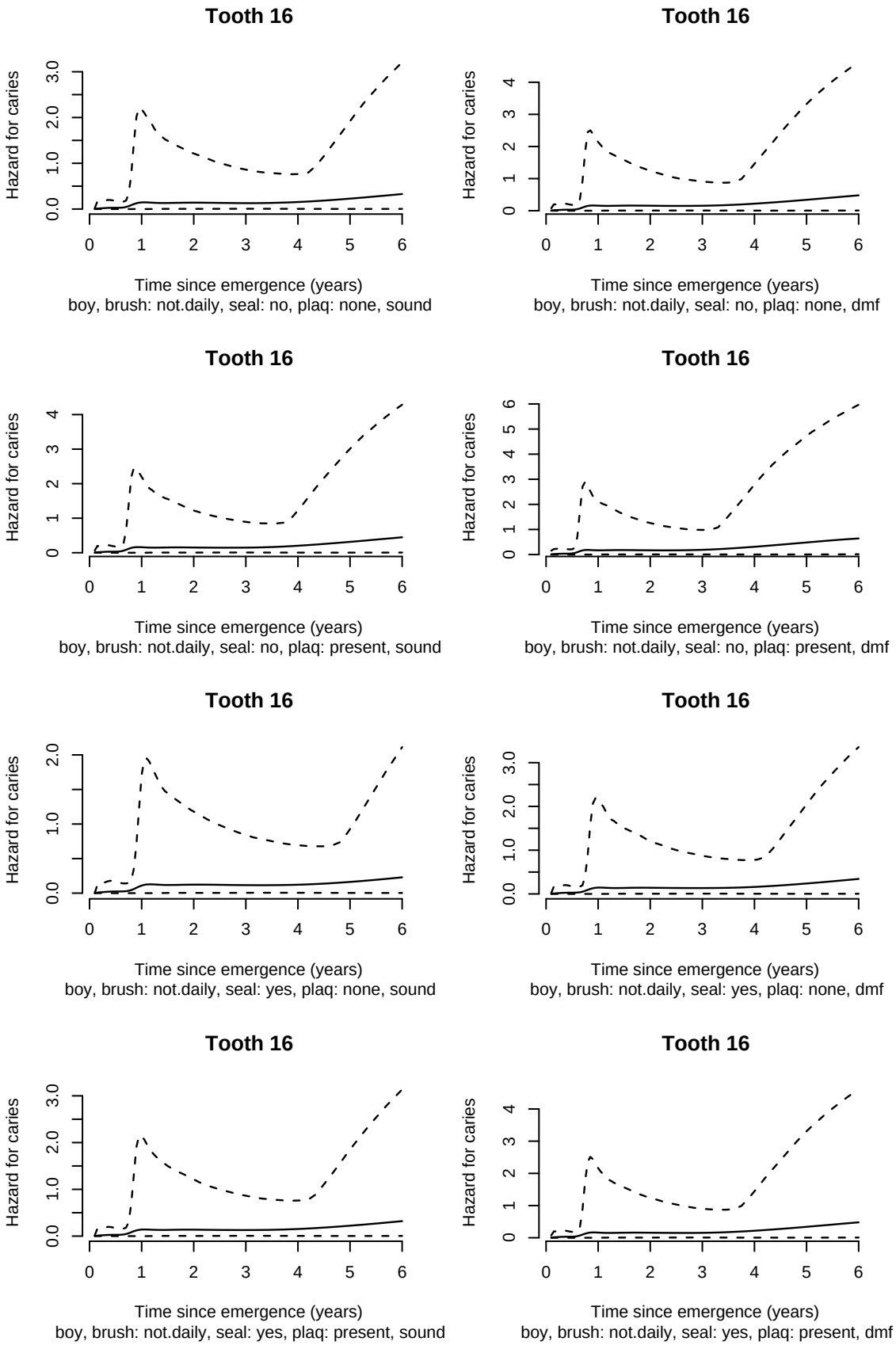


Figure 2: Predictive hazard curves (posterior predictive means and 95% pointwise credible regions) for caries for tooth 16 of boys.

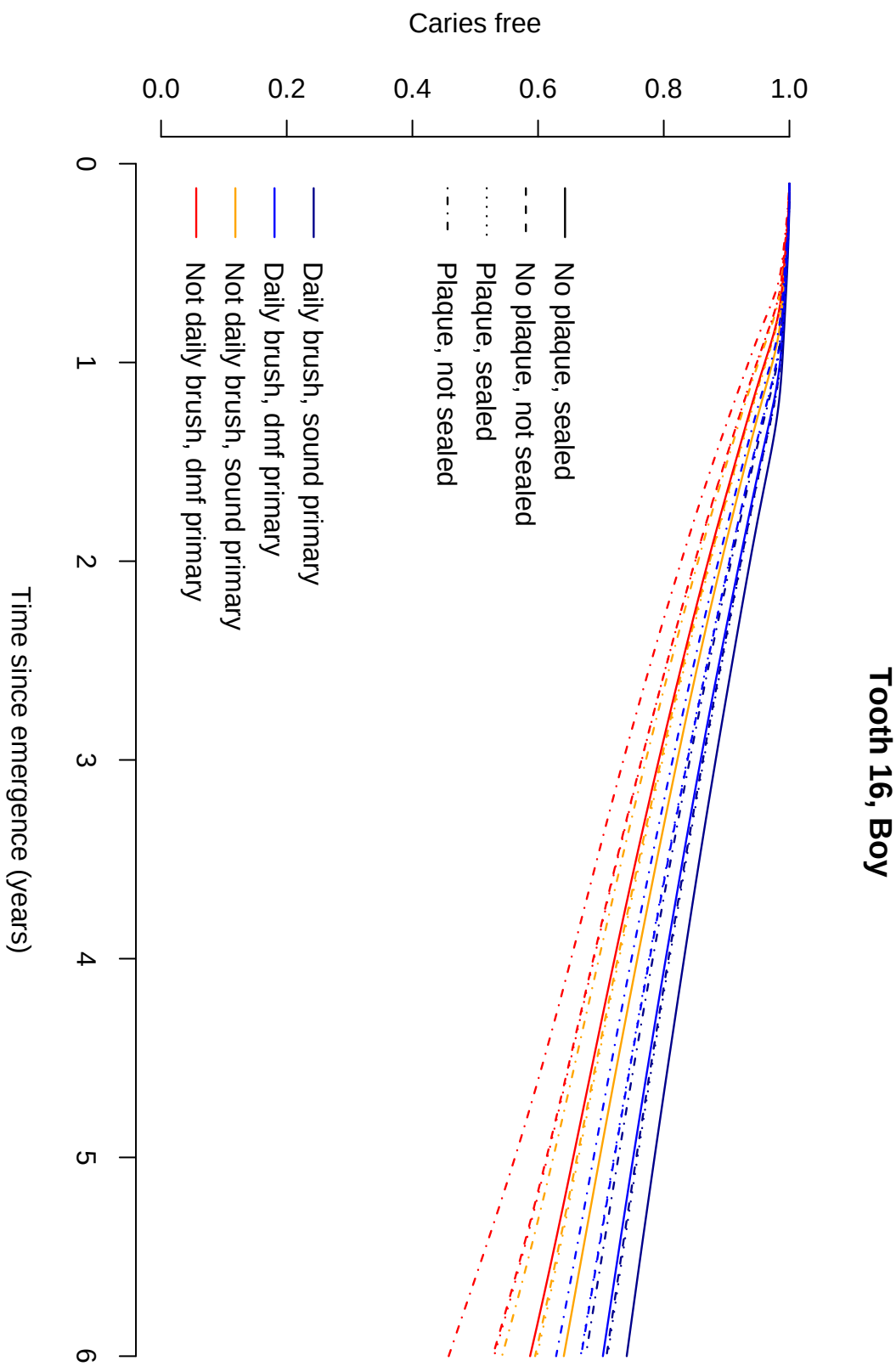


Figure 3: Predictive survivor curves (posterior predictive means) for caries for tooth 16 of boys.

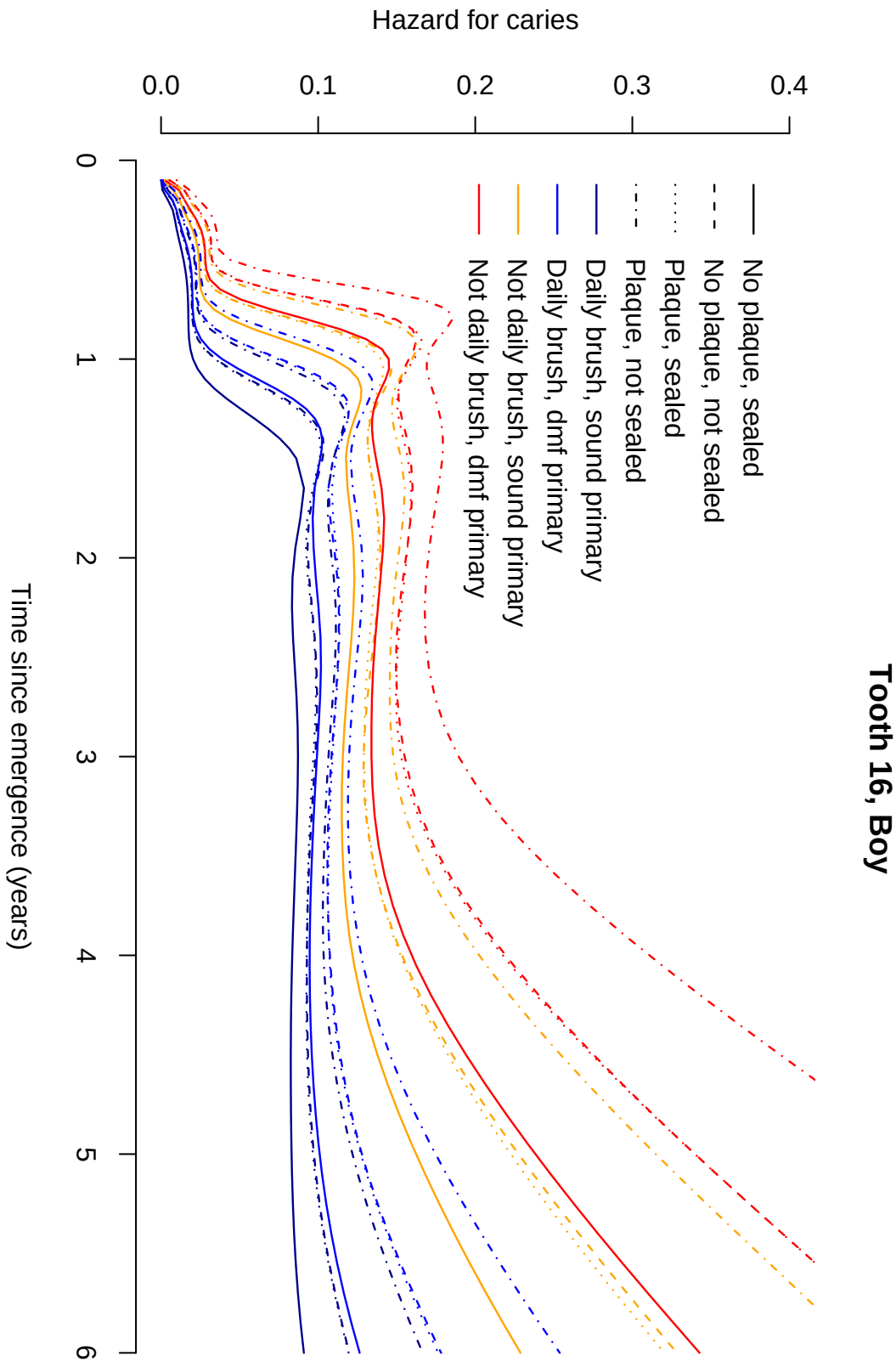


Figure 4: Predictive hazard curves (posterior predictive means) for caries for tooth 16 of boys.

7 Posterior predictive error and random effects densities – function `bayesGspline`

In this section we compute the estimates of the densities that were smoothed, store them in files and plot them. Computed estimates are stored in the directory `chaindir.CS`.

7.1 Compute and store the results

• Specify whether some sampled values are to be skipped and how often the progress information will be print on the screen.

```
> skip <- 0
> nwrite <- 5000
```

• Predictive density for the error term in the emergence part of the model ($g_\zeta(\zeta)$).

```
> grid.emerg <- seq(0.35, 0.55, length = 100)
> dens.emerg <- bayesGspline(chaindir.CS, grid1 = grid.emerg, skip = skip,
+     by = 1, nwrite = nwrite, extens = "", extens.adjust = "_b",
+     only.aver = TRUE, version = 30)
> sink(paste(chaindir.CS, "/densEmerg.dat", sep = ""), append = FALSE)
> cat(dens.emerg$grid, "\n", sep = "  ")
> cat(dens.emerg$average, "\n", sep = "  ")
> sink()
```

• Predictive density for the random effects in the emergence part of the model ($g_d(d)$).

```
> grid.random.emerg <- seq(-0.5, 0.5, length = 100)
> dens.random.emerg <- bayesGspline(chaindir.CS, grid1 = grid.random.emerg,
+     skip = skip, by = 1, nwrite = nwrite, extens = "_b", extens.adjust = "_b",
+     only.aver = TRUE, version = 31)
> sink(paste(chaindir.CS, "/densRandEmerg.dat", sep = ""), append = FALSE)
> cat(dens.random.emerg$grid, "\n", sep = "  ")
> cat(dens.random.emerg$average, "\n", sep = "  ")
> sink()
```

• Predictive density for the error term in the caries part of the model ($g_\varepsilon(\varepsilon)$).

```
> grid.caries <- seq(-1, 4, length = 100)
> dens.caries <- bayesGspline(chaindir.CS, grid1 = grid.caries,
+     skip = skip, by = 1, nwrite = nwrite, extens = "_2", extens.adjust = "_b2",
+     only.aver = TRUE, version = 30)
> sink(paste(chaindir.CS, "/densCaries.dat", sep = ""), append = FALSE)
> cat(dens.caries$grid, "\n", sep = "  ")
> cat(dens.caries$average, "\n", sep = "  ")
> sink()
```

• Predictive density for the random effects in the caries part of the model ($g_b(b)$).

```
> grid.random.caries <- seq(-2, 1.5, length = 100)
> dens.random.caries <- bayesGspline(chaindir.CS, grid1 = grid.random.caries,
+     skip = skip, by = 1, nwrite = nwrite, extens = "_b2", extens.adjust = "_b2",
+     only.aver = TRUE, version = 31)
> sink(paste(chaindir.CS, "/densRandCaries.dat", sep = ""), append = FALSE)
> cat(dens.random.caries$grid, "\n", sep = "  ")
> cat(dens.random.caries$average, "\n", sep = "  ")
> sink()
```

7.2 Read the results and plot them

The plots are stored in the directory `figuredir`.

• Some parameters for the plots.

```
> pdfwidth <- 9
> pdfheight <- 6
```

• Read computed densities from files.

```
> dens.emerg <- list(grid = scan(paste(chaindir.CS, "/densEmerg.dat",
+     sep = ""), nlines = 1), average = scan(paste(chaindir.CS,
+     "/densEmerg.dat", sep = ""), skip = 1, nlines = 1))
> dens.random.emerg <- list(grid = scan(paste(chaindir.CS, "/densRandEmerg.dat",
+     sep = ""), nlines = 1), average = scan(paste(chaindir.CS,
+     "/densRandEmerg.dat", sep = ""), skip = 1, nlines = 1))
> dens.caries <- list(grid = scan(paste(chaindir.CS, "/densCaries.dat",
+     sep = ""), nlines = 1), average = scan(paste(chaindir.CS,
+     "/densCaries.dat", sep = ""), skip = 1, nlines = 1))
> dens.random.caries <- list(grid = scan(paste(chaindir.CS, "/densRandCaries.dat",
+     sep = ""), nlines = 1), average = scan(paste(chaindir.CS,
+     "/densRandCaries.dat", sep = ""), skip = 1, nlines = 1))
```

• Plot the computed densities (see Figure 5).

```
> pdf(paste(figuredir, "err_randomDens_CS.pdf", sep = ""), width = pdfwidth,
+     height = pdfheight)
> par(mfcol = c(2, 2), bty = "n")
> plot(dens.emerg$grid, dens.emerg$average, type = "l", main = "Error, emergence",
+     xlab = expression(zeta), ylab = expression(g(zeta)))
> plot(dens.random.emerg$grid, dens.random.emerg$average, type = "l",
+     main = "Random intcpt, emergence", xlab = "d", ylab = "g(d)")
> plot(dens.caries$grid, dens.caries$average, type = "l", main = "Error, caries",
+     xlab = expression(epsilon), ylab = expression(g(epsilon)))
> plot(dens.random.caries$grid, dens.random.caries$average, type = "l",
+     main = "Random intcpt, caries", xlab = "b", ylab = "g(b)")
> dev.off()
```

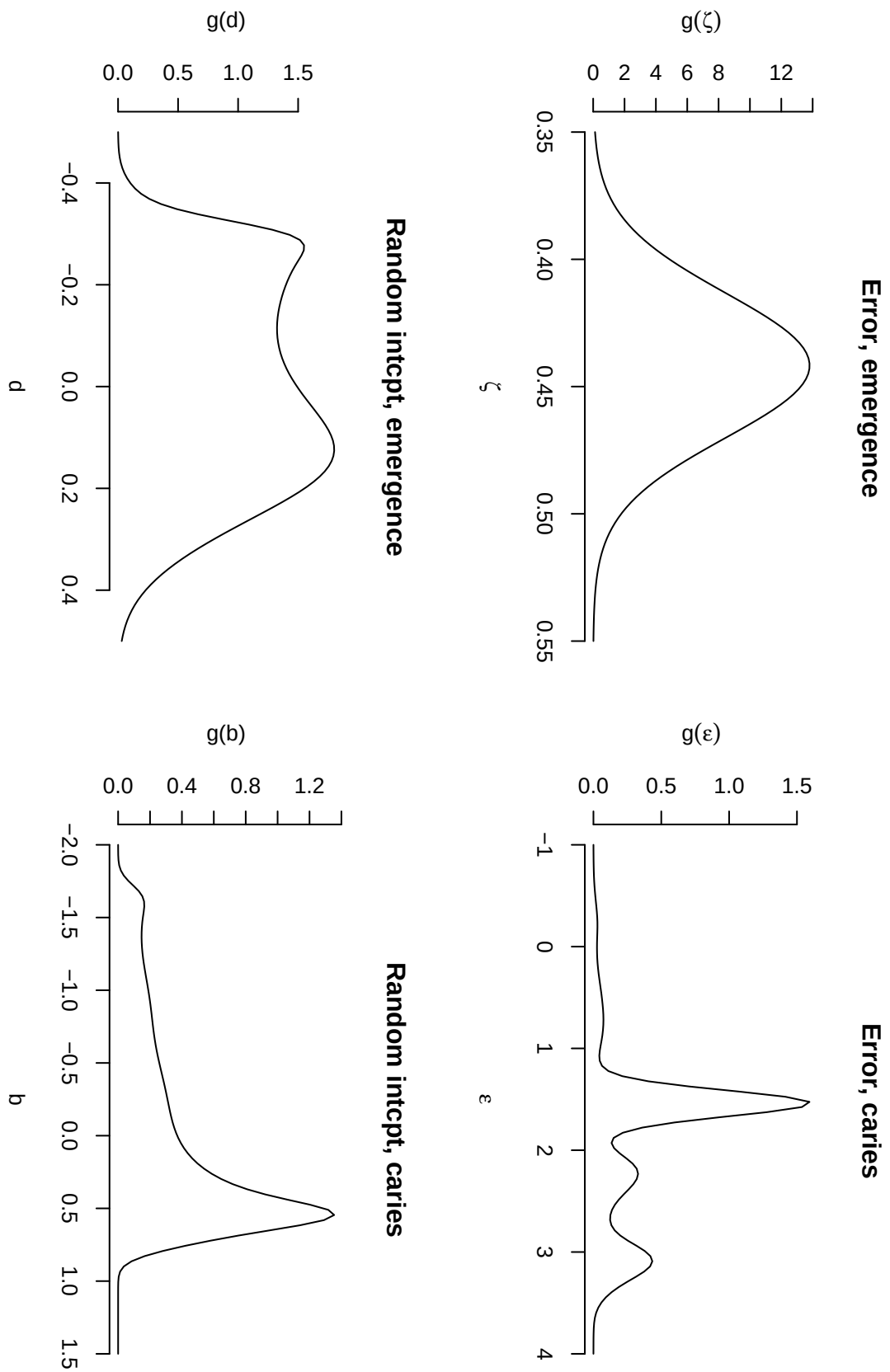


Figure 5: Estimate of the densities.

8 Read sampled chains

In this section we read the sampled chains. It is assumed that the chains are stored in the directory `chaindir.CS`.

• Numbers of regression parameters for the emergence and caries parts of the model.

```
> nbeta.emerg.CS <- 4
> nbeta.caries.CS <- 8
```

• Read the indeces of stored iterations and show the beginning using the standard function `str`.

```
> itersInd <- read.table(paste(chaindir.CS, "/iteration.sim", sep = ""),
+     header = TRUE)[, 1]
> nlines <- length(itersInd)
> str(itersInd)
```

```
 int [1:100000] 1000001 1000002 1000003 1000004 1000005 1000006 1000007 1000008 1000009 1000010 ...
```

8.1 Chains for the emergence part of the model

We read the sampled chains from the files `mixmoment.sim`, `lambda.sim`, `logposter.sim`, `beta.sim`, `mixmoment_b.sim`, `lambda_b.sim` and `logposter_b.sim` and store them (or their transformations) in the data frames baseline.emerg.CS, beta.emerg.CS, logposter.emerg.CS, logposter.b.emerg.CS, lambda.emerg.CS.

The overall intercept (variable `intercept`) is defined as the sum of the mean of g_ζ and the mean of g_d.

```
> mixmoment.emerg.CS <- matrix(scan(paste(chaindir.CS, "/mixmoment.sim",
+     sep = ""), skip = 1, nlines = nlines), ncol = 3, byrow = TRUE)
> lambda.emerg.CS <- matrix(scan(paste(chaindir.CS, "/lambda.sim",
+     sep = ""), skip = 1, nlines = nlines), ncol = 1, byrow = TRUE)
> logposter.emerg.CS <- matrix(scan(paste(chaindir.CS, "/logposter.sim",
+     sep = ""), skip = 1, nlines = nlines), ncol = 3, byrow = TRUE)
> beta.emerg.CS <- matrix(scan(paste(chaindir.CS, "/beta.sim",
+     sep = ""), skip = 1, nlines = nlines), ncol = nbeta.emerg.CS,
+     byrow = TRUE)
> mixmoment.b.emerg.CS <- matrix(scan(paste(chaindir.CS, "/mixmoment_b.sim",
+     sep = ""), skip = 1, nlines = nlines), ncol = 3, byrow = TRUE)
> lambda.b.emerg.CS <- matrix(scan(paste(chaindir.CS, "/lambda_b.sim",
+     sep = ""), skip = 1, nlines = nlines), ncol = 1, byrow = TRUE)
> logposter.b.emerg.CS <- matrix(scan(paste(chaindir.CS, "/logposter_b.sim",
+     sep = ""), skip = 1, nlines = nlines), ncol = 3, byrow = TRUE)
> colnames(mixmoment.emerg.CS) <- scan(paste(chaindir.CS, "/mixmoment.sim",
+     sep = ""), what = "character", nlines = 1)
> colnames(lambda.emerg.CS) <- scan(paste(chaindir.CS, "/lambda.sim",
+     sep = ""), what = "character", nlines = 1)
> colnames(logposter.emerg.CS) <- scan(paste(chaindir.CS, "/logposter.sim",
+     sep = ""), what = "character", nlines = 1)
> colnames(beta.emerg.CS) <- scan(paste(chaindir.CS, "/beta.sim",
+     sep = ""), what = "character", nlines = 1)
> colnames(mixmoment.b.emerg.CS) <- scan(paste(chaindir.CS, "/mixmoment_b.sim",
+     sep = ""), what = "character", nlines = 1)
> colnames(lambda.b.emerg.CS) <- scan(paste(chaindir.CS, "/lambda_b.sim",
+     sep = ""), what = "character", nlines = 1)
> colnames(logposter.b.emerg.CS) <- scan(paste(chaindir.CS, "/logposter_b.sim",
+     sep = ""), what = "character", nlines = 1)
> scale <- sqrt(mixmoment.emerg.CS[, "D.1.1"])
> scale.b <- sqrt(mixmoment.b.emerg.CS[, "D.1.1"])
```

```
> intercept <- mixmoment.emerg.CS[, "Mean.1"] + mixmoment.b.emerg.CS[,
+     "Mean.1"]
> baseline.emerg.CS <- data.frame(Intcpt.16 = intercept, Intcpt.26 = intercept +
+     beta.emerg.CS[, "Tooth26"], Intcpt.36 = intercept + beta.emerg.CS[,
+     "Tooth36"], Intcpt.46 = intercept + beta.emerg.CS[, "Tooth46"],
+     Scale = scale, Scale.b = scale.b)
> beta.emerg.CS <- as.data.frame(beta.emerg.CS)
> logposter.emerg.CS <- as.data.frame(logposter.emerg.CS)
> logposter.b.emerg.CS <- as.data.frame(logposter.b.emerg.CS)
> lambda.emerg.CS <- data.frame(lambda = lambda.emerg.CS[, 1],
+     lambda.b = lambda.b.emerg.CS[, 1])
> rm(list = c("intercept", "scale", "scale.b", "mixmoment.emerg.CS",
+     "mixmoment.b.emerg.CS", "lambda.b.emerg.CS"))
```

• Show the beginning of each read and kept quantity using the standard function `str`.

```
> str(baseline.emerg.CS)

'data.frame':        100000 obs. of  6 variables:
 $ Intcpt.16: num  0.438 0.438 0.437 0.439 0.438 ...
 $ Intcpt.26: num  0.431 0.431 0.429 0.432 0.434 ...
 $ Intcpt.36: num  0.439 0.440 0.440 0.441 0.441 ...
 $ Intcpt.46: num  0.433 0.434 0.434 0.435 0.437 ...
 $ Scale    : num  0.0298 0.0300 0.0311 0.0304 0.0300 ...
 $ Scale.b  : num  0.203 0.204 0.203 0.201 0.199 ...

> str(beta.emerg.CS)

'data.frame':        100000 obs. of  4 variables:
 $ Tooth26: num  -0.00730 -0.00721 -0.00762 -0.00647 -0.00480 ...
 $ Tooth36: num  0.00139 0.00219 0.00321 0.00260 0.00244 ...
 $ Tooth46: num  -0.00447 -0.00395 -0.00331 -0.00340 -0.00117 ...
 $ Girl   : num  -0.0225 -0.0233 -0.0234 -0.0243 -0.0258 ...

> str(logposter.emerg.CS)

'data.frame':        100000 obs. of  3 variables:
 $ loglik : num  2447 2429 2484 2414 2697 ...
 $ penalty: num  -278 -268 -270 -292 -243 ...
 $ logprw : num  -761 -855 -866 -824 -686 ...

> str(logposter.b.emerg.CS)

'data.frame':        100000 obs. of  3 variables:
 $ loglik : num  628 729 718 687 682 ...
 $ penalty: num  -0.0702 -0.0664 -0.0297 -0.0240 -0.0271 ...
 $ logprw : num  -10243 -10243 -10247 -10269 -10272 ...

> str(lambda.emerg.CS)

'data.frame':        100000 obs. of  2 variables:
 $ lambda  : num  0.0409 0.0785 0.0804 0.0531 0.0737 ...
 $ lambda.b: num  187 249 587 439 590 ...
```

8.2 Chains for the caries part of the model

We read the sampled chains from the files `mixmoment_2.sim`, `lambda_2.sim`, `logposter_2.sim`, `beta_2.sim`, `mixmoment_b2.sim`, `lambda_b2.sim` and `logposter_b2.sim` and store them (or their transformations) in the data frames baseline.caries.CS, beta.caries.CS, logposter.caries.CS, logposter.b.caries.CS, lambda.caries.CS.

The overall intercept (variable `intercept`) is defined as the sum of the mean of g_ε and the mean of g_b.

```
> mixmoment.caries.CS <- matrix(scan(paste(chaindir.CS, "/mixmoment_2.sim",
+     sep = ""), skip = 1, nlines = nlines), ncol = 3, byrow = TRUE)
> lambda.caries.CS <- matrix(scan(paste(chaindir.CS, "/lambda_2.sim",
+     sep = ""), skip = 1, nlines = nlines), ncol = 1, byrow = TRUE)
> logposter.caries.CS <- matrix(scan(paste(chaindir.CS, "/logposter_2.sim",
+     sep = ""), skip = 1, nlines = nlines), ncol = 3, byrow = TRUE)
> beta.caries.CS <- matrix(scan(paste(chaindir.CS, "/beta_2.sim",
+     sep = ""), skip = 1, nlines = nlines), ncol = nbeta.caries.CS,
+     byrow = TRUE)
> mixmoment.b.caries.CS <- matrix(scan(paste(chaindir.CS, "/mixmoment_b2.sim",
+     sep = ""), skip = 1, nlines = nlines), ncol = 3, byrow = TRUE)
> lambda.b.caries.CS <- matrix(scan(paste(chaindir.CS, "/lambda_b2.sim",
+     sep = ""), skip = 1, nlines = nlines), ncol = 1, byrow = TRUE)
> logposter.b.caries.CS <- matrix(scan(paste(chaindir.CS, "/logposter_b2.sim",
+     sep = ""), skip = 1, nlines = nlines), ncol = 3, byrow = TRUE)
> colnames(mixmoment.caries.CS) <- scan(paste(chaindir.CS, "/mixmoment_2.sim",
+     sep = ""), what = "character", nlines = 1)
> colnames(lambda.caries.CS) <- scan(paste(chaindir.CS, "/lambda_2.sim",
+     sep = ""), what = "character", nlines = 1)
> colnames(logposter.caries.CS) <- scan(paste(chaindir.CS, "/logposter_2.sim",
+     sep = ""), what = "character", nlines = 1)
> colnames(beta.caries.CS) <- scan(paste(chaindir.CS, "/beta_2.sim",
+     sep = ""), what = "character", nlines = 1)
> colnames(mixmoment.b.caries.CS) <- scan(paste(chaindir.CS, "/mixmoment_b2.sim",
+     sep = ""), what = "character", nlines = 1)
> colnames(lambda.b.caries.CS) <- scan(paste(chaindir.CS, "/lambda_b2.sim",
+     sep = ""), what = "character", nlines = 1)
> colnames(logposter.b.caries.CS) <- scan(paste(chaindir.CS, "/logposter_b2.sim",
+     sep = ""), what = "character", nlines = 1)
> scale <- sqrt(mixmoment.caries.CS[, "D.1.1"])
> scale.b <- sqrt(mixmoment.b.caries.CS[, "D.1.1"])
> intercept <- mixmoment.caries.CS[, "Mean.1"] + mixmoment.b.caries.CS[,
+     "Mean.1"]
> baseline.caries.CS <- data.frame(Intcpt.16 = intercept, Intcpt.26 = intercept +
+     beta.caries.CS[, "Tooth26"], Intcpt.36 = intercept + beta.caries.CS[,
+     "Tooth36"], Intcpt.46 = intercept + beta.caries.CS[, "Tooth46"],
+     Scale = scale, Scale.b = scale.b)
> beta.caries.CS <- as.data.frame(beta.caries.CS)
> logposter.caries.CS <- as.data.frame(logposter.caries.CS)
> logposter.b.caries.CS <- as.data.frame(logposter.b.caries.CS)
> lambda.caries.CS <- data.frame(lambda = lambda.caries.CS[, 1],
+     lambda.b = lambda.b.caries.CS[, 1])
> rm(list = c("intercept", "scale", "scale.b", "mixmoment.caries.CS",
+     "mixmoment.b.caries.CS", "lambda.b.caries.CS"))
```

• Show the beginning of each read and kept quantity using the standard function `str`.

```
> str(baseline.caries.CS)
```

```
'data.frame':        100000 obs. of  6 variables:
 $ Intcpt.16: num  2.00 2.02 2.00 2.01 2.01 ...
```

```
 $ Intcpt.26: num  1.96 1.98 1.97 1.98 1.99 ...
 $ Intcpt.36: num  1.99 2.01 2.00 2.00 2.00 ...
 $ Intcpt.46: num  1.98 2.01 1.99 1.99 2.00 ...
 $ Scale    : num  0.873 0.868 0.866 0.866 0.866 ...
 $ Scale.b  : num  0.669 0.665 0.664 0.668 0.664 ...
```

> str(beta.caries.CS)

```
'data.frame':        100000 obs. of  8 variables:
 $ Tooth26: num  -0.0385 -0.0352 -0.0337 -0.0256 -0.0160 ...
 $ Tooth36: num  -0.00395 -0.00696 -0.00616 -0.00408 -0.00482 ...
 $ Tooth46: num  -0.0168 -0.0087 -0.0127 -0.0213 -0.0129 ...
 $ Girl   : num  -0.0214 -0.0196 -0.0233 -0.0276 -0.0316 ...
 $ Brush  : num  0.308 0.309 0.313 0.313 0.304 ...
 $ Plaque : num  -0.137 -0.140 -0.143 -0.142 -0.148 ...
 $ Seal   : num  0.0455 0.0413 0.0390 0.0353 0.0325 ...
 $ Prim5  : num  -0.156 -0.158 -0.154 -0.153 -0.145 ...
```

> str(logposter.caries.CS)

```
'data.frame':        100000 obs. of  3 variables:
 $ loglik : num  2361 2404 2429 2348 2370 ...
 $ penalty: num  -40.1 -61.3 -59.9 -58.3 -46.3 ...
 $ logprw : num  -30156 -30125 -30318 -30012 -30110 ...
```

> str(logposter.b.caries.CS)

```
'data.frame':        100000 obs. of  3 variables:
 $ loglik : num  703 653 583 721 659 ...
 $ penalty: num  -0.0326 -0.0308 -0.0288 -0.0154 -0.0334 ...
 $ logprw : num  -11057 -11035 -11044 -11055 -11047 ...
```

> str(lambda.caries.CS)

```
'data.frame':        100000 obs. of  2 variables:
 $ lambda  : num  0.440 0.280 0.197 0.322 0.396 ...
 $ lambda.b: num  390 452 715 594 259 ...
```

9 Summary for some parameters – function `give.summary`

In this section we show how to compute some summary for the sampled chains.

• Compute the summary.

```
> summ.baseline.emerg.CS <- give.summary(baseline.emerg.CS)
> summ.baseline.caries.CS <- give.summary(baseline.caries.CS)
> summ.beta.emerg.CS <- give.summary(beta.emerg.CS)
> summ.beta.caries.CS <- give.summary(beta.caries.CS)
> summ.lambda.emerg.CS <- give.summary(lambda.emerg.CS)
> summ.lambda.caries.CS <- give.summary(lambda.caries.CS)
```

• Round the summary.

```
> rsumm.baseline.emerg.CS <- round(summ.baseline.emerg.CS, dig = 4)
> rsumm.baseline.caries.CS <- round(summ.baseline.caries.CS, dig = 4)
> rsumm.beta.emerg.CS <- round(summ.beta.emerg.CS, dig = 4)
> rsumm.beta.caries.CS <- round(summ.beta.caries.CS, dig = 4)
> rsumm.lambda.emerg.CS <- round(summ.lambda.emerg.CS, dig = 4)
> rsumm.lambda.caries.CS <- round(summ.lambda.caries.CS, dig = 4)
```

• Print the summary for the emergence part of the model.

```
> print(rsumm.beta.emerg.CS)
```

```
        Tooth26 Tooth36 Tooth46    Girl
means   -0.0030  0.0015  0.0022 -0.0231
50%     -0.0029  0.0015  0.0022 -0.0231
2.5%    -0.0112 -0.0066 -0.0060 -0.0393
97.5%    0.0050  0.0096  0.0104 -0.0071
p.value  0.4635  0.7228  0.5960  0.0076
```

```
> print(rsumm.baseline.emerg.CS)
```

```
        Intcpt.16 Intcpt.26 Intcpt.36 Intcpt.46  Scale Scale.b
means      0.4416    0.4386    0.4431    0.4438 0.0293  0.1992
50%        0.4416    0.4387    0.4431    0.4439 0.0293  0.1986
2.5%       0.4274    0.4241    0.4289    0.4292 0.0250  0.1910
97.5%      0.4558    0.4527    0.4570    0.4578 0.0336  0.2097
p.value    0.0000    0.0000    0.0000    0.0000 0.0000  0.0000
```

```
> print(rsumm.lambda.emerg.CS)
```

```
        lambda  lambda.b
means   0.0463  427.5539
50%     0.0387  371.0814
2.5%    0.0076   70.0313
97.5%   0.1296 1100.4140
p.value 0.0000    0.0000
```

• Print the summary for the caries part of the model.

```
> print(rsumm.beta.caries.CS)
```

```
        Tooth26 Tooth36 Tooth46    Girl  Brush  Plaque   Seal   Prim5
means   -0.0065 -0.0089 -0.0158 -0.0713 0.3368 -0.1153 0.1184 -0.1402
```

```
50%     -0.0064 -0.0088 -0.0157 -0.0715 0.3367 -0.1141 0.1187 -0.1396
2.5%    -0.0385 -0.0444 -0.0515 -0.1549 0.2328 -0.1706 0.0596 -0.1927
97.5%    0.0242  0.0268  0.0194  0.0089 0.4361 -0.0674 0.1781 -0.0909
p.value  0.6879  0.6335  0.3752  0.0848 0.0000  0.0000 0.0004  0.0000
```

```
> print(rsumm.baseline.caries.CS)
```

```
        Intcpt.16 Intcpt.26 Intcpt.36 Intcpt.46  Scale Scale.b
means      1.9226    1.9161    1.9137    1.9068 0.7685  0.6724
50%        1.9201    1.9135    1.9119    1.9043 0.7669  0.6723
2.5%       1.8100    1.8047    1.8005    1.7963 0.7116  0.6136
97.5%      2.0594    2.0511    2.0451    2.0376 0.8335  0.7340
p.value    0.0000    0.0000    0.0000    0.0000 0.0000  0.0000
```

```
> print(rsumm.lambda.caries.CS)
```

```
        lambda  lambda.b
means   0.2759  621.8005
50%     0.1383  561.1106
2.5%    0.0145  151.4273
97.5%   1.4949 1438.4868
p.value 0.0000    0.0000
```

10 Posterior densities and traceplots for some parameters – functions densplot2 and traceplot2

In this section we show how to draw traceplots and estimates of the posterior densities for some parameters.

10.1 Posterior densities

These can be drawn using the following code.

```
> plfun <- "densplot2"
> plname <- "dens"
> pdfwidth <- 9
> pdfheight <- 6
> tt <- c(16, 26, 36, 46)
```

• Effect of Tooth and Girl on the emergence (see Figure 6).

```
> pdf(paste(figuredir, plname, "_emergBeta.pdf", sep = ""), width = pdfwidth,
+     height = pdfheight)
> par(mfrow = c(2, 2), bty = "n")
> eval(call(plfun, mcmc(beta.emerg.CS[, "Tooth26"]), main = "Tooth26 (emerg)"))
> title(sub = paste("Tooth26 = ", rsumm.beta.emerg.CS[1, "Tooth26"],
+     " (", rsumm.beta.emerg.CS[3, "Tooth26"], ", ", rsumm.beta.emerg.CS[4,
+         "Tooth26"], ")", sep = ""))
> eval(call(plfun, mcmc(beta.emerg.CS[, "Tooth36"]), main = "Tooth36 (emerg)"))
> title(sub = paste("Tooth36 = ", rsumm.beta.emerg.CS[1, "Tooth36"],
+     " (", rsumm.beta.emerg.CS[3, "Tooth36"], ", ", rsumm.beta.emerg.CS[4,
+         "Tooth36"], ")", sep = ""))
> eval(call(plfun, mcmc(beta.emerg.CS[, "Tooth46"]), main = "Tooth46 (emerg)"))
> title(sub = paste("Tooth46 = ", rsumm.beta.emerg.CS[1, "Tooth46"],
+     " (", rsumm.beta.emerg.CS[3, "Tooth46"], ", ", rsumm.beta.emerg.CS[4,
+         "Tooth46"], ")", sep = ""))
> eval(call(plfun, mcmc(beta.emerg.CS[, "Girl"]), main = "Girl (emerg)"))
> title(sub = paste("Girl = ", rsumm.beta.emerg.CS[1, "Girl"],
+     " (", rsumm.beta.emerg.CS[3, "Girl"], ", ", rsumm.beta.emerg.CS[4,
+         "Girl"], ")", sep = ""))
> dev.off()
```

• Intercepts (overall intercept plus the effect of the tooth) for the emergence process (see Figure 7).

```
> pdf(paste(figuredir, plname, "_emergIntcpt.pdf", sep = ""), width = pdfwidth,
+     height = pdfheight)
> par(mfrow = c(2, 2), bty = "n")
> eval(call(plfun, mcmc(baseline.emerg.CS[, "Intcpt.16"]), main = "Intcpt (emerg), tooth 16"))
> title(sub = paste("Intercept = ", rsumm.baseline.emerg.CS[1,
+     "Intcpt.16"], " (", rsumm.baseline.emerg.CS[3, "Intcpt.16"],
+     ", ", rsumm.baseline.emerg.CS[4, "Intcpt.16"], ")", sep = ""))
> eval(call(plfun, mcmc(baseline.emerg.CS[, "Intcpt.26"]), main = "Intcpt (emerg), tooth 26"))
> title(sub = paste("Intercept = ", rsumm.baseline.emerg.CS[1,
+     "Intcpt.26"], " (", rsumm.baseline.emerg.CS[3, "Intcpt.26"],
+     ", ", rsumm.baseline.emerg.CS[4, "Intcpt.26"], ")", sep = ""))
> eval(call(plfun, mcmc(baseline.emerg.CS[, "Intcpt.36"]), main = "Intcpt (emerg), tooth 36"))
> title(sub = paste("Intercept = ", rsumm.baseline.emerg.CS[1,
+     "Intcpt.36"], " (", rsumm.baseline.emerg.CS[3, "Intcpt.36"],
+     ", ", rsumm.baseline.emerg.CS[4, "Intcpt.36"], ")", sep = ""))
> eval(call(plfun, mcmc(baseline.emerg.CS[, "Intcpt.46"]), main = "Intcpt (emerg), tooth 46"))
```

```
> title(sub = paste("Intercept = ", rsumm.baseline.emerg.CS[1,
+     "Intcpt.46"], " (", rsumm.baseline.emerg.CS[3, "Intcpt.46"],
+     ", ", rsumm.baseline.emerg.CS[4, "Intcpt.46"], ")", sep = ""))
> dev.off()
```

• Scale (sd(ζ)) and Scale.b (sd(d)) for the emergence process and smoothing parameters λ^{ζ} and λ^{d} for the emergence (see Figure 8).

```
> pdf(paste(figuredir, plname, "_emergScaleLambda.pdf", sep = ""),
+     width = pdfwidth, height = pdfheight)
> par(mfrow = c(2, 2), bty = "n")
> eval(call(plfun, mcmc(baseline.emerg.CS[, "Scale"]), main = "Scale (emerg)"))
> title(sub = paste("Intercept = ", rsumm.baseline.emerg.CS[1,
+     "Scale"], " (", rsumm.baseline.emerg.CS[3, "Scale"], ", ",
+     rsumm.baseline.emerg.CS[4, "Scale"], ")", sep = ""))
> eval(call(plfun, mcmc(baseline.emerg.CS[, "Scale.b"]), main = "Scale of b (emerg)"))
> title(sub = paste("Intercept = ", rsumm.baseline.emerg.CS[1,
+     "Scale.b"], " (", rsumm.baseline.emerg.CS[3, "Scale.b"],
+     ", ", rsumm.baseline.emerg.CS[4, "Scale.b"], ")", sep = ""))
> eval(call(plfun, mcmc(lambda.emerg.CS[, "lambda"]), main = "Lambda (emerg)"))
> title(sub = paste("lambda = ", rsumm.lambda.emerg.CS[1, "lambda"],
+     " (", rsumm.lambda.emerg.CS[3, "lambda"], ", ", rsumm.lambda.emerg.CS[4,
+         "lambda"], ")", sep = ""))
> eval(call(plfun, mcmc(lambda.emerg.CS[, "lambda.b"]), main = "Lambda of b (emerg)"))
> title(sub = paste("lambda.b = ", rsumm.lambda.emerg.CS[1, "lambda.b"],
+     " (", rsumm.lambda.emerg.CS[3, "lambda.b"], ", ", rsumm.lambda.emerg.CS[4,
+         "lambda.b"], ")", sep = ""))
> dev.off()
```

• Effect of Tooth and Girl on the caries (see Figure 9).

```
> pdf(paste(figuredir, plname, "_cariesBeta1.pdf", sep = ""), width = pdfwidth,
+     height = pdfheight)
> par(mfrow = c(2, 2), bty = "n")
> eval(call(plfun, mcmc(beta.caries.CS[, "Tooth26"]), main = "Tooth26 (caries)"))
> title(sub = paste("Tooth26 = ", rsumm.beta.caries.CS[1, "Tooth26"],
+     " (", rsumm.beta.caries.CS[3, "Tooth26"], ", ", rsumm.beta.caries.CS[4,
+         "Tooth26"], ")", sep = ""))
> eval(call(plfun, mcmc(beta.caries.CS[, "Tooth36"]), main = "Tooth36 (caries)"))
> title(sub = paste("Tooth36 = ", rsumm.beta.caries.CS[1, "Tooth36"],
+     " (", rsumm.beta.caries.CS[3, "Tooth36"], ", ", rsumm.beta.caries.CS[4,
+         "Tooth36"], ")", sep = ""))
> eval(call(plfun, mcmc(beta.caries.CS[, "Tooth46"]), main = "Tooth46 (caries)"))
> title(sub = paste("Tooth46 = ", rsumm.beta.caries.CS[1, "Tooth46"],
+     " (", rsumm.beta.caries.CS[3, "Tooth46"], ", ", rsumm.beta.caries.CS[4,
+         "Tooth46"], ")", sep = ""))
> eval(call(plfun, mcmc(beta.caries.CS[, "Girl"]), main = "Girl (caries)"))
> title(sub = paste("Girl = ", rsumm.beta.caries.CS[1, "Girl"],
+     " (", rsumm.beta.caries.CS[3, "Girl"], ", ", rsumm.beta.caries.CS[4,
+         "Girl"], ")", sep = ""))
> dev.off()
```

• Effect of Brush, Seal, Plaque, Prim5 on the caries (see Figure 10).

```
> pdf(paste(figuredir, plname, "_cariesBeta2.pdf", sep = ""), width = pdfwidth,
+     height = pdfheight)
> par(mfrow = c(2, 2), bty = "n")
> eval(call(plfun, mcmc(beta.caries.CS[, "Brush"]), main = "Brush (caries)"))
> title(sub = paste("Brush = ", rsumm.beta.caries.CS[1, "Brush"],
```

```
+     " (", rsumm.beta.caries.CS[3, "Brush"], ", ", rsumm.beta.caries.CS[4,
+         "Brush"], ")", sep = ""))
> eval(call(plfun, mcmc(beta.caries.CS[, "Seal"]), main = "Seal (caries)"))
> title(sub = paste("Seal = ", rsumm.beta.caries.CS[1, "Seal"],
+     " (", rsumm.beta.caries.CS[3, "Seal"], ", ", rsumm.beta.caries.CS[4,
+         "Seal"], ")", sep = ""))
> eval(call(plfun, mcmc(beta.caries.CS[, "Plaque"]), main = "Plaque (caries)"))
> title(sub = paste("Plaque = ", rsumm.beta.caries.CS[1, "Plaque"],
+     " (", rsumm.beta.caries.CS[3, "Plaque"], ", ", rsumm.beta.caries.CS[4,
+         "Plaque"], ")", sep = ""))
> eval(call(plfun, mcmc(beta.caries.CS[, "Prim5"]), main = "Prim5 (caries)"))
> title(sub = paste("Prim5 = ", rsumm.beta.caries.CS[1, "Prim5"],
+     " (", rsumm.beta.caries.CS[3, "Prim5"], ", ", rsumm.beta.caries.CS[4,
+         "Prim5"], ")", sep = ""))
> dev.off()
```

• Intercepts (overall intercept plus the effect of the tooth) for the caries process (see Figure 11).

```
> pdf(paste(figuredir, plname, "_cariesIntcpt.pdf", sep = ""),
+     width = pdfwidth, height = pdfheight)
> par(mfrow = c(2, 2), bty = "n")
> eval(call(plfun, mcmc(baseline.caries.CS[, "Intcpt.16"]), main = "Intcpt (caries), tooth 16"))
> title(sub = paste("Intercept = ", rsumm.baseline.caries.CS[1,
+     "Intcpt.16"], " (", rsumm.baseline.caries.CS[3, "Intcpt.16"],
+     ", ", rsumm.baseline.caries.CS[4, "Intcpt.16"], ")", sep = ""))
> eval(call(plfun, mcmc(baseline.caries.CS[, "Intcpt.26"]), main = "Intcpt (caries), tooth 26"))
> title(sub = paste("Intercept = ", rsumm.baseline.caries.CS[1,
+     "Intcpt.26"], " (", rsumm.baseline.caries.CS[3, "Intcpt.26"],
+     ", ", rsumm.baseline.caries.CS[4, "Intcpt.26"], ")", sep = ""))
> eval(call(plfun, mcmc(baseline.caries.CS[, "Intcpt.36"]), main = "Intcpt (caries), tooth 36"))
> title(sub = paste("Intercept = ", rsumm.baseline.caries.CS[1,
+     "Intcpt.36"], " (", rsumm.baseline.caries.CS[3, "Intcpt.36"],
+     ", ", rsumm.baseline.caries.CS[4, "Intcpt.36"], ")", sep = ""))
> eval(call(plfun, mcmc(baseline.caries.CS[, "Intcpt.46"]), main = "Intcpt (caries), tooth 46"))
> title(sub = paste("Intercept = ", rsumm.baseline.caries.CS[1,
+     "Intcpt.46"], " (", rsumm.baseline.caries.CS[3, "Intcpt.46"],
+     ", ", rsumm.baseline.caries.CS[4, "Intcpt.46"], ")", sep = ""))
> dev.off()
```

• Scale (sd(ε)) and Scale.b (sd(b)) for the caries process and smoothing parameters λ^{ε} and λ^{b} for the caries (see Figure 12).

```
> pdf(paste(figuredir, plname, "_cariesScaleLambda.pdf", sep = ""),
+     width = pdfwidth, height = pdfheight)
> par(mfrow = c(2, 2), bty = "n")
> eval(call(plfun, mcmc(baseline.caries.CS[, "Scale"]), main = "Scale (caries)"))
> title(sub = paste("Intercept = ", rsumm.baseline.caries.CS[1,
+     "Scale"], " (", rsumm.baseline.caries.CS[3, "Scale"], ", ",
+     rsumm.baseline.caries.CS[4, "Scale"], ")", sep = ""))
> eval(call(plfun, mcmc(baseline.caries.CS[, "Scale.b"]), main = "Scale of b (caries)"))
> title(sub = paste("Intercept = ", rsumm.baseline.caries.CS[1,
+     "Scale.b"], " (", rsumm.baseline.caries.CS[3, "Scale.b"],
+     ", ", rsumm.baseline.caries.CS[4, "Scale.b"], ")", sep = ""))
> eval(call(plfun, mcmc(lambda.caries.CS[, "lambda"]), main = "Lambda (caries)"))
> title(sub = paste("lambda = ", rsumm.lambda.caries.CS[1, "lambda"],
+     " (", rsumm.lambda.caries.CS[3, "lambda"], ", ", rsumm.lambda.caries.CS[4,
+         "lambda"], ")", sep = ""))
> eval(call(plfun, mcmc(lambda.caries.CS[, "lambda.b"]), main = "Lambda of b (caries)"))
> title(sub = paste("lambda.b = ", rsumm.lambda.caries.CS[1, "lambda.b"],
```

```
+     " (", rsumm.lambda.caries.CS[3, "lambda.b"], ", ", rsumm.lambda.caries.CS[4,
+         "lambda.b"], ")", sep = ""))
> dev.off()
```

10.2 Traceplots

These can be drawn using the same code with the only change at the beginning.

```
> plfun <- "traceplot2"
> plname <- "trace"
```

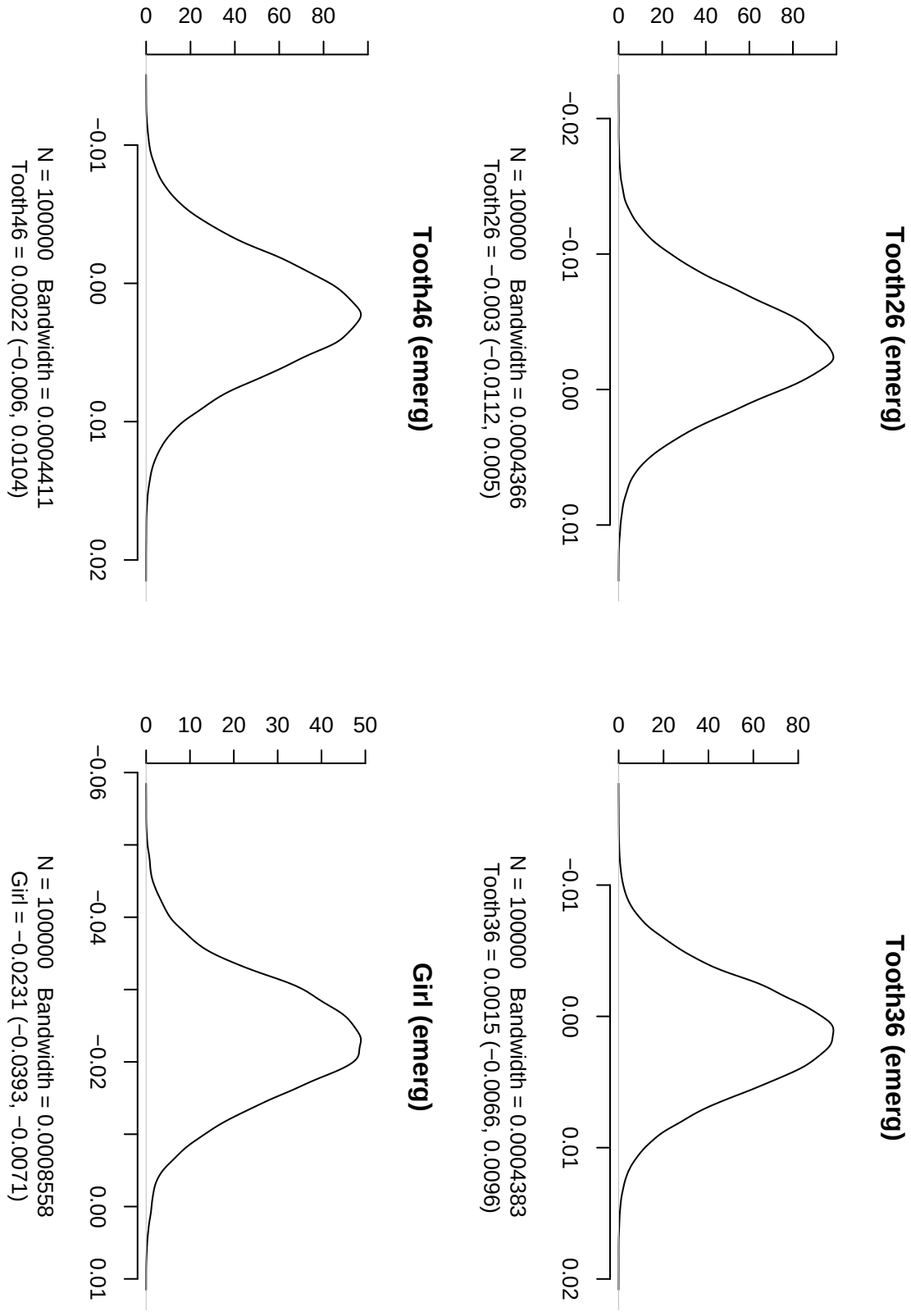


Figure 6: Estimate of the posterior distribution.

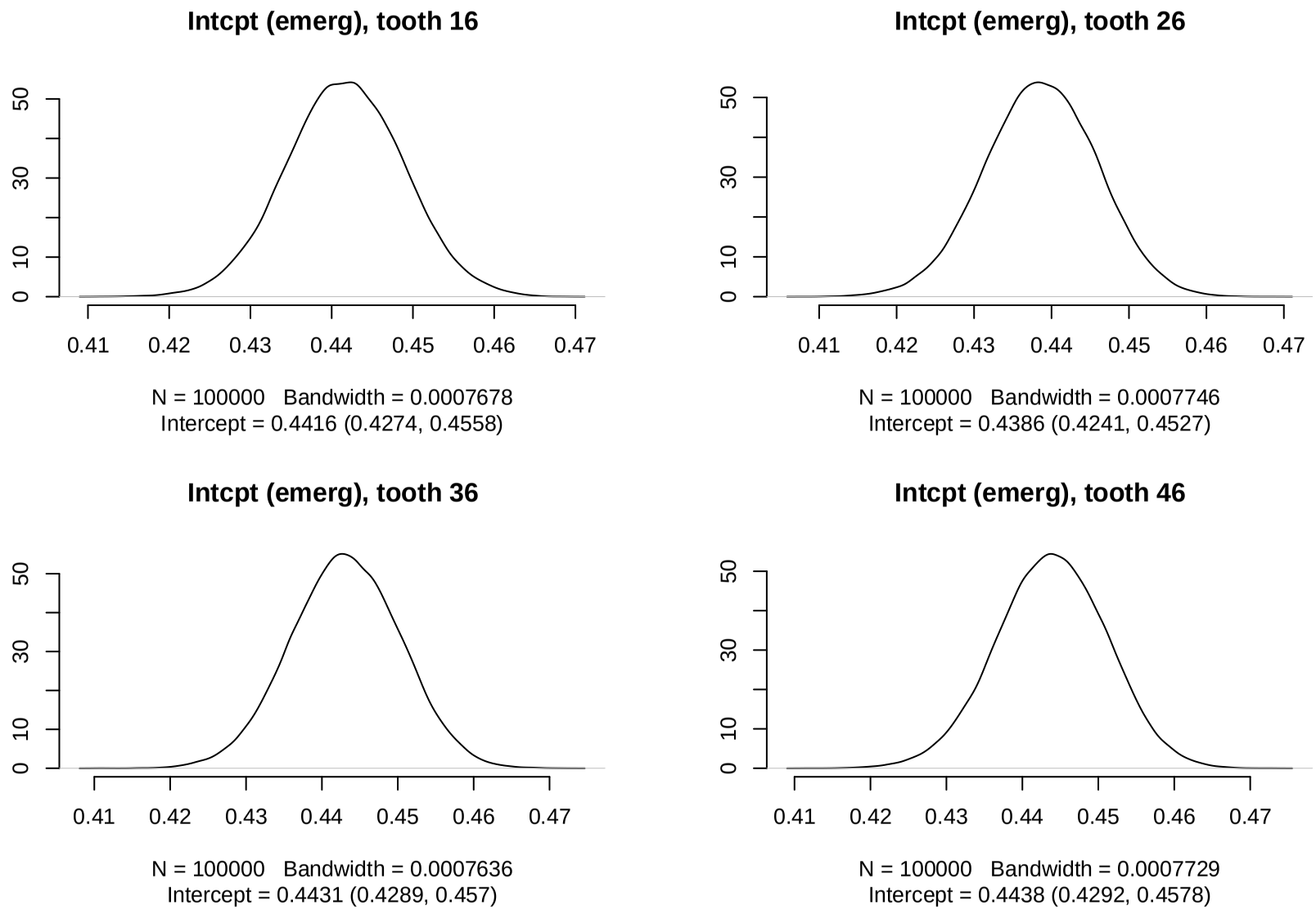
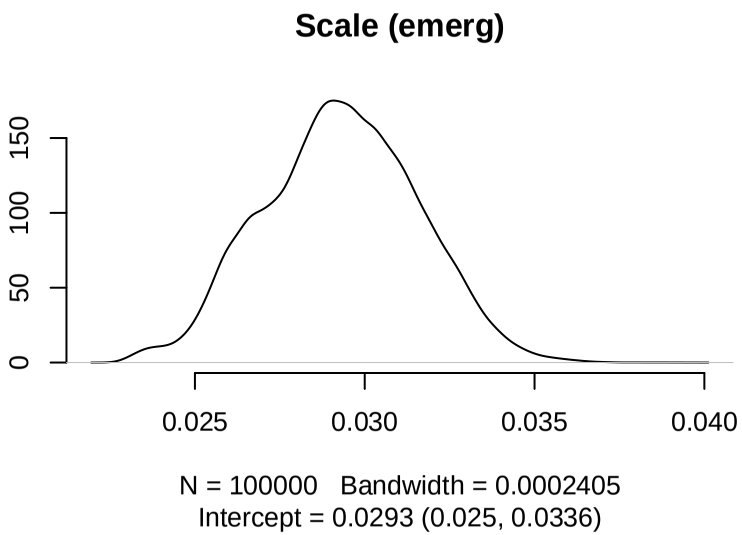
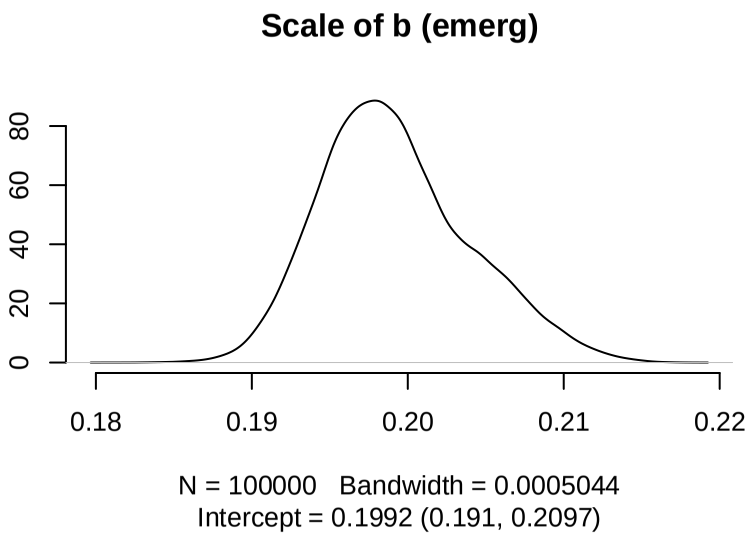
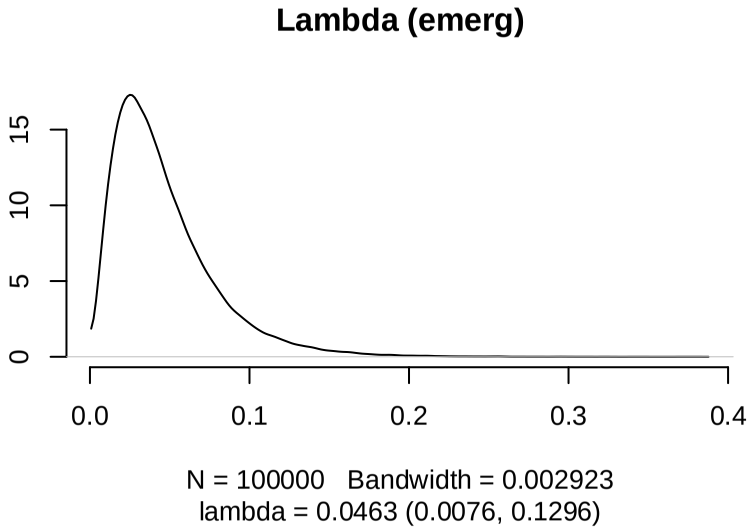


Figure 7: Estimate of the posterior distribution.







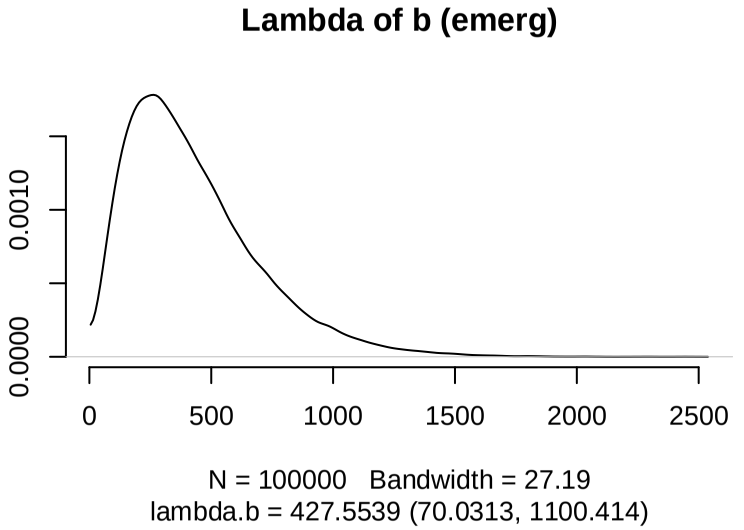


Figure 8: Estimate of the posterior distribution.

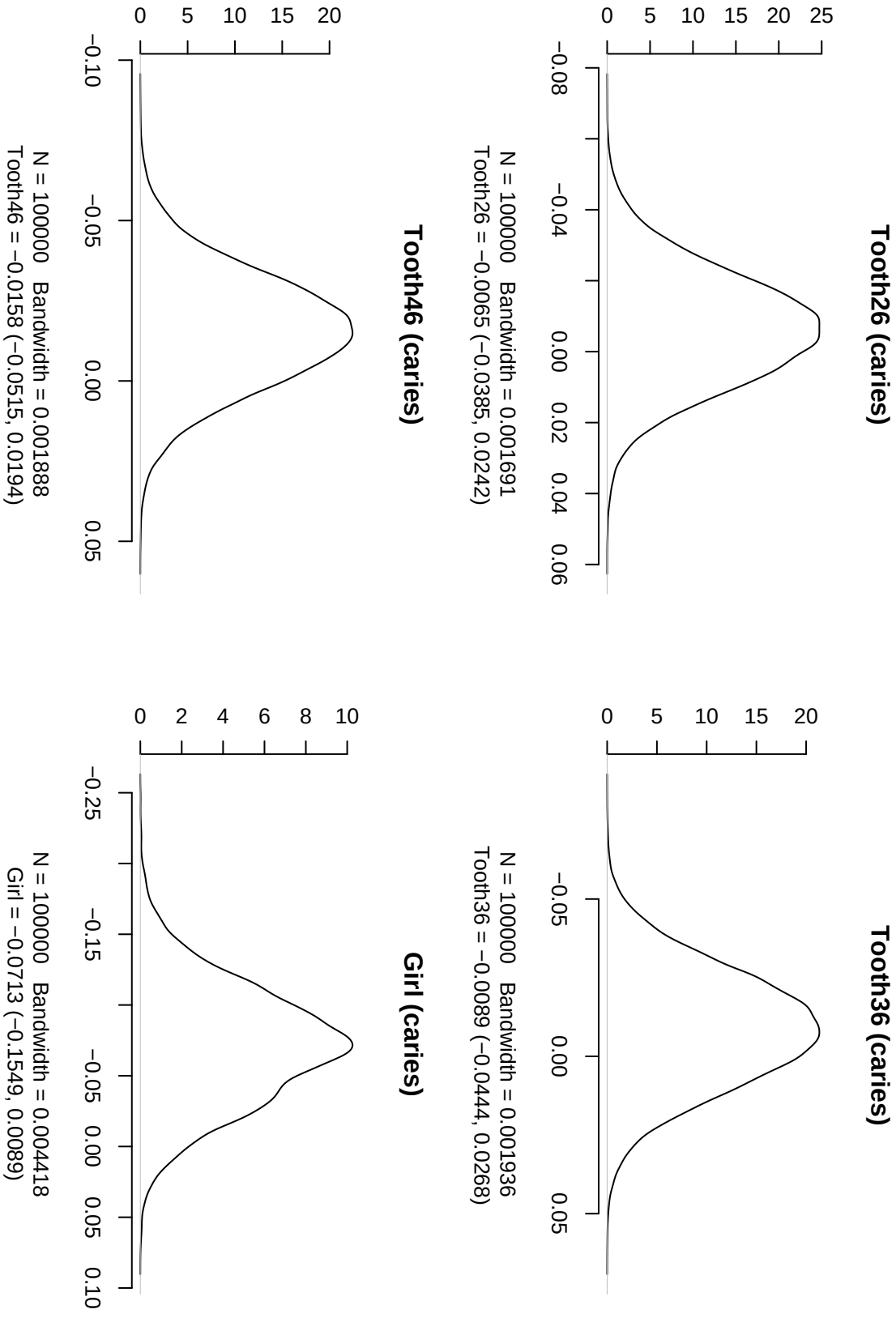


Figure 9: Estimate of the posterior distribution.

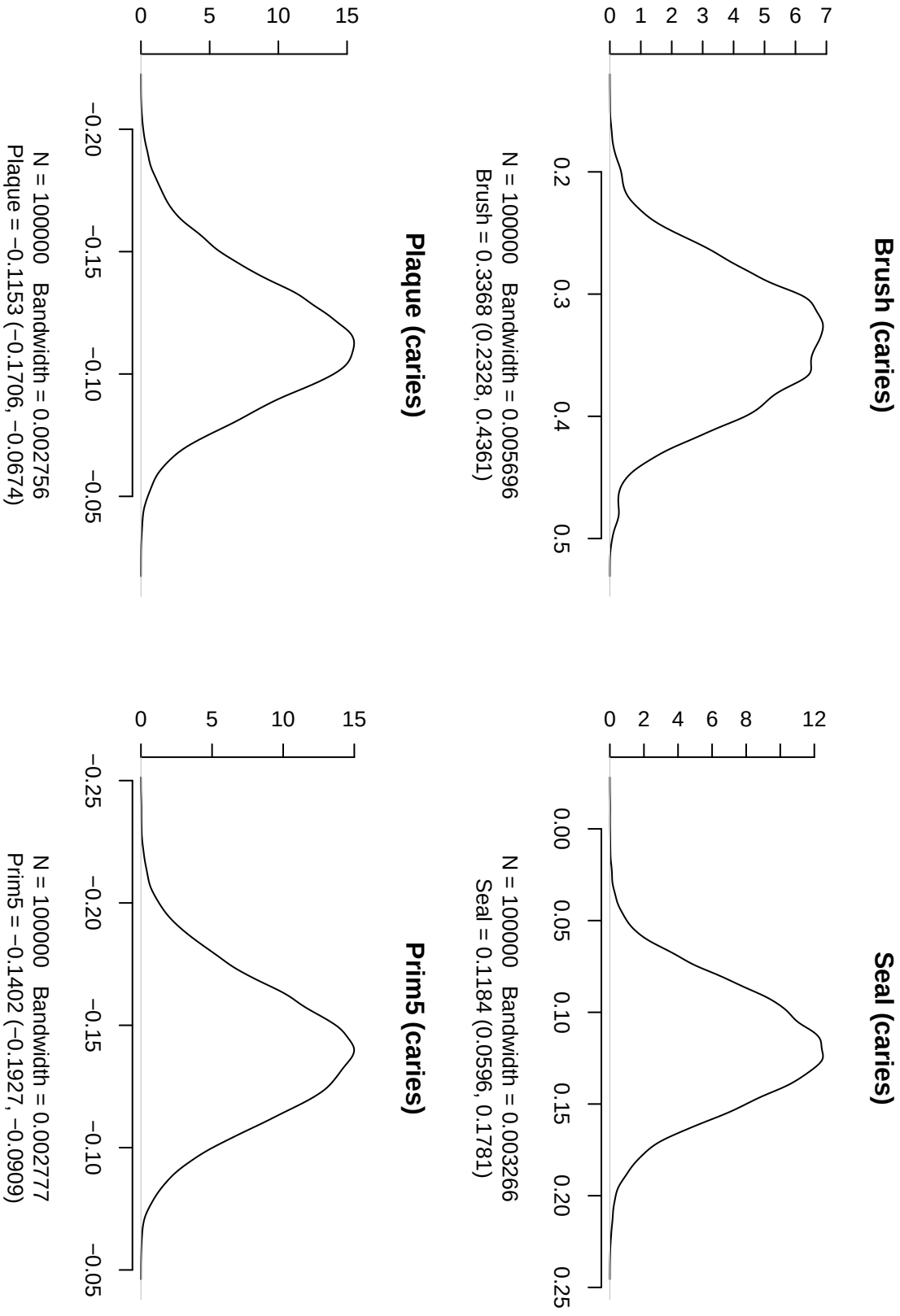
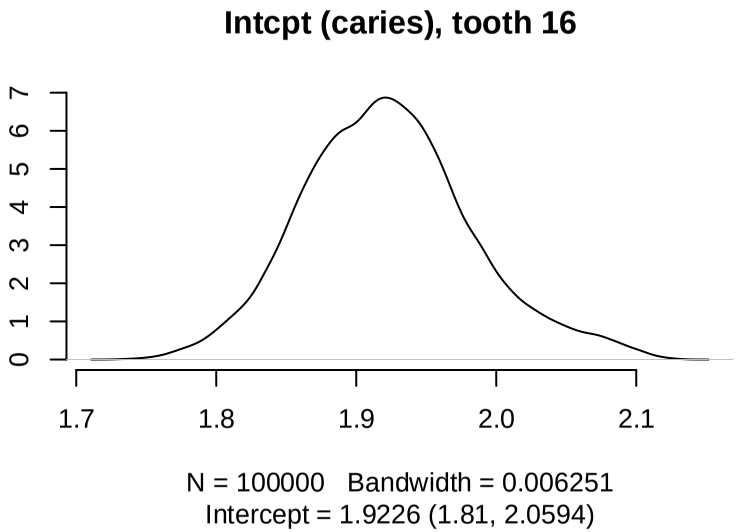
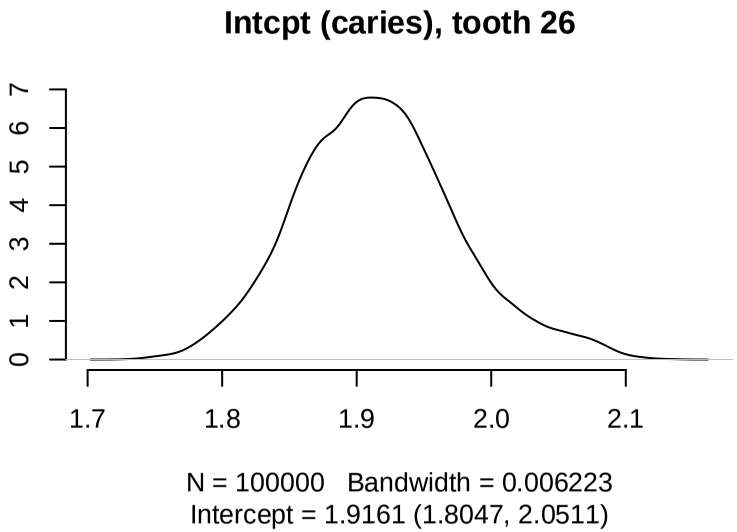
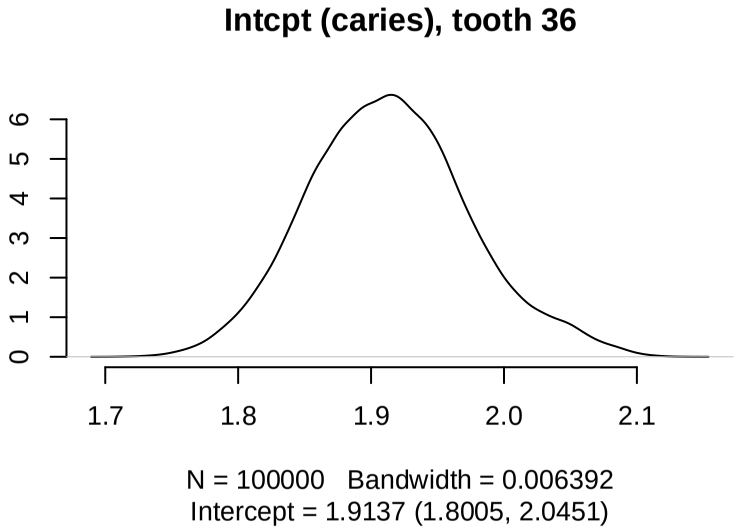


Figure 10: Estimate of the posterior distribution.







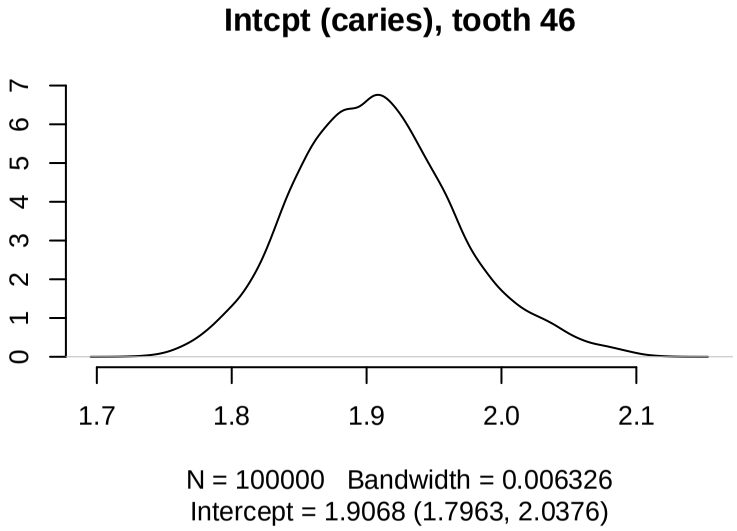
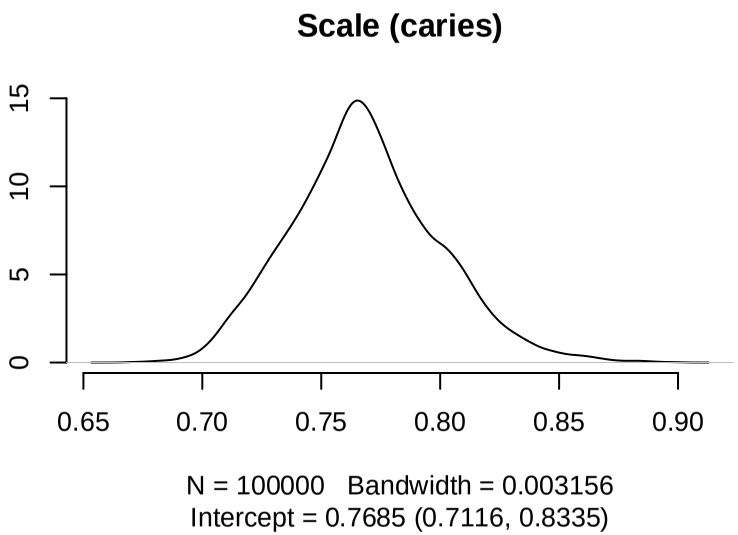
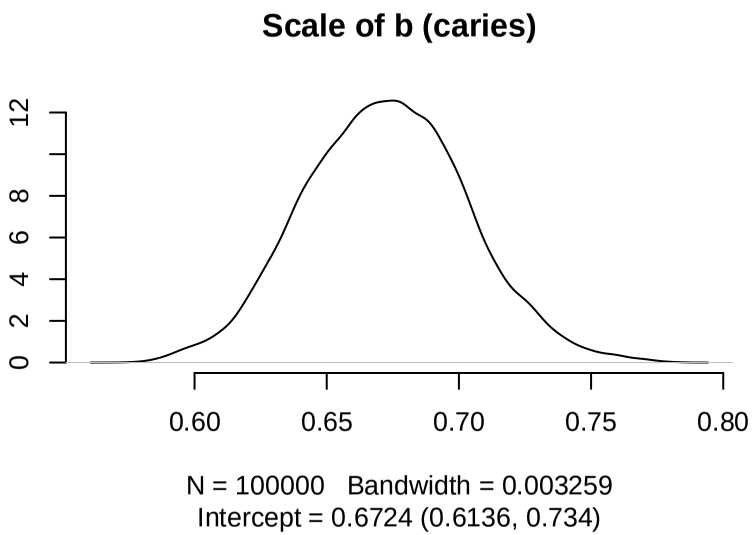
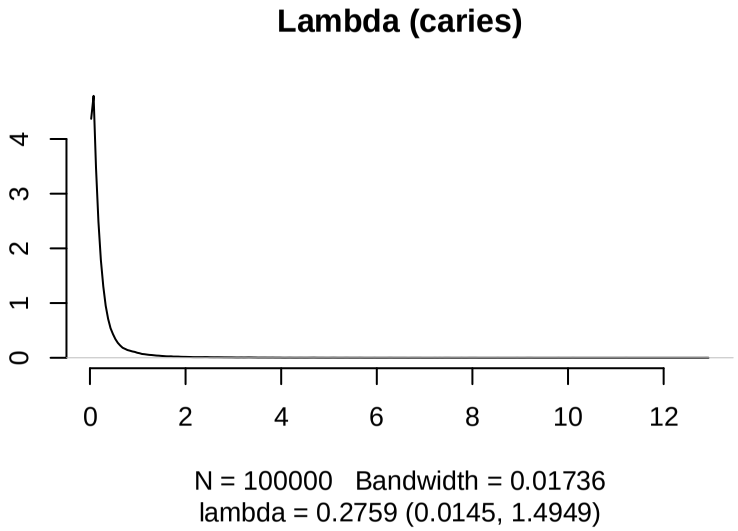


Figure 11: Estimate of the posterior distribution.







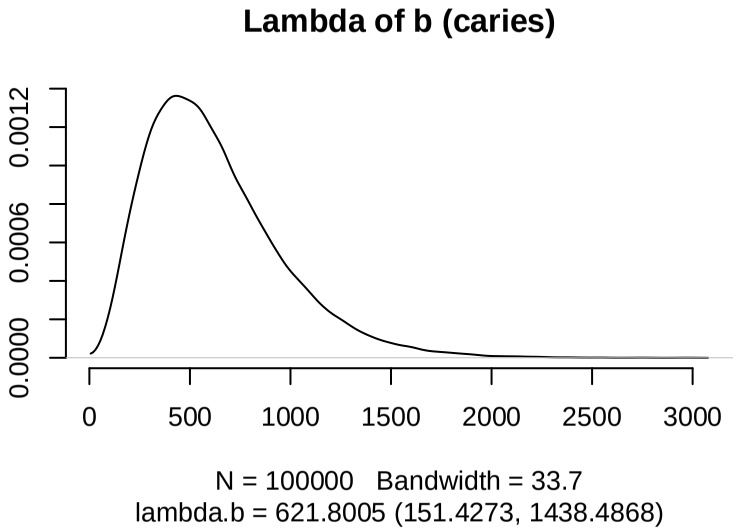


Figure 12: Estimate of the posterior distribution.