2017 SISG Bayesian Statistics for Genetics
R Notes: Generalized Linear Modeling

Jon Wakefield
Departments of Statistics and Biostatistics, University of Washington

2017-09-05

Overview

In this set of notes a number of generalized linear models (GLMs) and generalized linear mixed models (GLMMs) will be fitted using Bayesian methods.

Two primary computational techniques will be illustrated:

– integrated nested Laplace approximation (INLA)

– Markov chain Monte Carlo (MCMC) using `Stan`

Case control example: Data

We analyze a case control example using logistic regression models, first using likelihood methods.

The data concern the numbers of cases (of the disease Leber Hereditary Optic Neuropathy) and controls as a function of genotype at a particular location (rs6767450).

```
x <- c(0, 1, 2)
# Case data for CC CT TT
y <- c(6, 8, 75)
# Control data for CC CT TT
z <- c(10, 66, 163)
```

Case control example: Likelihood analysis

We fit the logistic regression model as a generalized linear model and then examine the estimate and an asymptotic (large sample) 95% confidence interval.

```
logitmod <- glm(cbind(y, z) ~ x, family = "binomial")
thetahat <- logitmod$coeff[2]  # Log odds ratio
thetahat
##         x
## 0.4787428
exp(thetahat)  # Odds ratio
##        x
## 1.614044
V <- vcov(logitmod)[2, 2]  # standard error^2
# Asymptotic confidence interval for odds ratio
exp(thetahat - 1.96 * sqrt(V))
##         x
## 0.9879159
exp(thetahat + 1.96 * sqrt(V))
##        x
## 2.637004
```

Case control example: Likelihood analysis

Now let's look at a likelihood ratio test of $H_0 : \theta = 0$ where θ is the log odds ratio associated with the genotype (multiplicative model).

```
logitmod
##
## Call:  glm(formula = cbind(y, z) ~ x, family = "binomial")
##
## Coefficients:
## (Intercept)            x
##     -1.8077       0.4787
##
## Degrees of Freedom: 2 Total (i.e. Null);  1 Residual
## Null Deviance:       15.01
## Residual Deviance: 10.99     AIC: 27.79
dev <- logitmod$null.deviance - logitmod$deviance
dev
## [1] 4.01874
pchisq(dev, df = logitmod$df.residual, lower.tail = F)
## [1] 0.04499731
```

So just significant at the 5% level.

FTO Example: Data

We reproduce the least squares analysis of the FTO data.

```
fto <- structure(list(y = c(2.14709237851534, 6.16728664844416,
    6.5287427751799, 13.7905616042756, 13.6590155436307,
    1.75906323176397, 6.77485810485697, 9.67664941025843,
    11.751562703307, 12.3892232256873, 9.723562369017,
    12.1796864728229, 14.8575188389164, 16.370600225645,
    27.7498618362862, 6.61013278196954, 11.3676194738021,
    17.9876724213706, 22.4424423901962, 26.687802642435),
    X = structure(c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1,
        2, 3, 4, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
        2, 3, 4, 5, 1, 2, 3, 4, 5), .Dim = c(20L, 4L),
        .Dimnames = list(NULL, c("", "xg", "xa", "")))),
    .Names = c("y", "X"))
```

FTO Example: Data and LS Fit

The `lm` function uses MLE, which is equivalent to ordinary least squares.

```
liny <- fto$y
linxg <- fto$X[, "xg"]
linxa <- fto$X[, "xa"]
linxint <- fto$X[, "xg"] * fto$X[, "xa"]
ftodf <- list(liny = liny, linxg = linxg, linxa = linxa,
    linxint = linxint)
ols.fit <- lm(liny ~ linxg + linxa + linxint, data = ftodf)
```

FTO Example: LS fit

```
summary(ols.fit)
##
## Call:
## lm(formula = liny ~ linxg + linxa + linxint, data = ftodf)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8008 -0.8844  0.2993  1.2270  2.4819
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.06822    1.42230  -0.048   0.9623
## linxg        2.94485    2.01143   1.464   0.1625
## linxa        2.84421    0.42884   6.632 5.76e-06 ***
## linxint      1.72948    0.60647   2.852   0.0115 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.918 on 16 degrees of freedom
## Multiple R-squared:  0.9393, Adjusted R-squared:  0.9279
## F-statistic: 82.55 on 3 and 16 DF,  p-value: 5.972e-10
```

INLA

Integrated nested Laplace approximation (INLA) is a technique for carrying out Bayesian computation.

It is not a standard `R` package and must be downloaded from the development website.

The `inla` function is the work horse.

```
# install.packages('INLA',
# repos='http://www.math.ntnu.no/inla/R/stable')
library(INLA)
# Data should be input to INLA as either a list or
# a dataframe
formula <- liny ~ linxg + linxa + linxint
lin.mod <- inla(formula, data = ftodf, family = "gaussian")
```

We might wonder, where are the priors?

We didn't specify any. . . but INLA has default choices.

FTO example via INLA: Lots of output available!

```
names(lin.mod)
##  [1] "names.fixed"                "summary.fixed"
##  [3] "marginals.fixed"            "summary.lincomb"
##  [5] "marginals.lincomb"          "size.lincomb"
##  [7] "summary.lincomb.derived"    "marginals.lincomb.derived"
##  [9] "size.lincomb.derived"       "mlik"
## [11] "cpo"                        "po"
## [13] "waic"                       "model.random"
## [15] "summary.random"             "marginals.random"
## [17] "size.random"                "summary.linear.predictor"
## [19] "marginals.linear.predictor" "summary.fitted.values"
## [21] "marginals.fitted.values"    "size.linear.predictor"
## [23] "summary.hyperpar"           "marginals.hyperpar"
## [25] "internal.summary.hyperpar"  "internal.marginals.hyperpar"
## [27] "offset.linear.predictor"    "model.spde2.blc"
## [29] "summary.spde2.blc"          "marginals.spde2.blc"
## [31] "size.spde2.blc"             "model.spde3.blc"
## [33] "summary.spde3.blc"          "marginals.spde3.blc"
## [35] "size.spde3.blc"             "logfile"
## [37] "misc"                       "dic"
## [39] "mode"                       "neffp"
## [41] "joint.hyper"                "nhyper"
## [43] "version"                    "Q"
## [45] "graph"                      "ok"
## [47] "cpu.used"                   "all.hyper"
## [49] ".args"                      "call"
## [51] "model.matrix"
```

FTO example: INLA analysis

The posterior means and posterior standard deviations are in very close agreement with the OLS fits presented earlier.

```
coef(ols.fit)
## (Intercept)       linxg       linxa     linxint
## -0.06821632  2.94485495  2.84420729  1.72947648
sqrt(diag(vcov(ols.fit)))
## (Intercept)       linxg       linxa     linxint
##   1.4222970   2.0114316   0.4288387   0.6064695
lin.mod$summary.fixed
##                    mean        sd 0.025quant    0.5quant 0.975quant
## (Intercept) -0.06162675 1.4255330 -2.8905368 -0.06207205   2.765482
## linxg        2.93325519 2.0135746 -1.0661222  2.93388526   6.922284
## linxa        2.84237279 0.4298886  1.9886025  2.84246901   3.694175
## linxint      1.73261904 0.6073436  0.5274574  1.73240953   2.937132
##                    mode          kld
## (Intercept) -0.0626946 1.490420e-11
## linxg        2.9351309 1.469269e-11
## linxa        2.8426701 1.493726e-11
## linxint      1.7321107 1.475964e-11
```

FTO example: INLA analysis

```
summary(lin.mod)
##
## Call:
## "inla(formula = formula, family = \"gaussian\", data = ftodf)"
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          0.9292          0.1732          0.1601          1.2624
##
## Fixed effects:
##                mean     sd 0.025quant 0.5quant 0.975quant    mode kld
## (Intercept) -0.0616 1.4255    -2.8905  -0.0621     2.7655 -0.0627   0
## linxg        2.9333 2.0136    -1.0661   2.9339     6.9223  2.9351   0
## linxa        2.8424 0.4299     1.9886   2.8425     3.6942  2.8427   0
## linxint      1.7326 0.6073     0.5275   1.7324     2.9371  1.7321   0
##
## The model has no random effects
##
## Model hyperparameters:
##                                                mean     sd 0.025quant 0.5quant
## Precision for the Gaussian observations 0.3047 0.0987     0.1412   0.2943
##                                                0.975quant  mode
## Precision for the Gaussian observations     0.5255 0.272
##
## Expected number of effective parameters(std dev): 3.995(0.0018)
## Number of equivalent replicates : 5.006
```

FTO Posterior marginals

We now examine the posterior marginal distributions.

The posterior marginal distribution for the vector of regression coefficients (including the intercept) is given below, and then we examine the posterior marginal on the precision, $1/\sigma_\epsilon$.

Check out the files that are written.

```
plot(lin.mod, plot.hyperparameter = FALSE, plot.fixed.effects =
    prefix = "linmodplot2", postscript = T)
plot(lin.mod, plot.hyperparameter = TRUE, plot.fixed.effects = F
    prefix = "linmodplot1", postscript = T)
```

FTO example via INLA

In order to carry out model checking we rerun the analysis, but now switch on a flag to obtain fitted values.

```
lin.mod <- inla(liny ~ linxg + linxa + linxint, data = ftodf,
    family = "gaussian", control.predictor = list(compute = TRUE))
fitted <- lin.mod$summary.fitted.values[, 1]
# Now extract the posterior median of the
# measurement error sd
sigmamed <- 1/sqrt(lin.mod$summary.hyperpar[, 4])
```

FTO: Residual analysis

With the fitted values we can examine the fit of the model. In particular:

- Normality of the errors (sample size is relatively small).
- Errors have constant variance (and are uncorrelated).

FTO Residual analysis

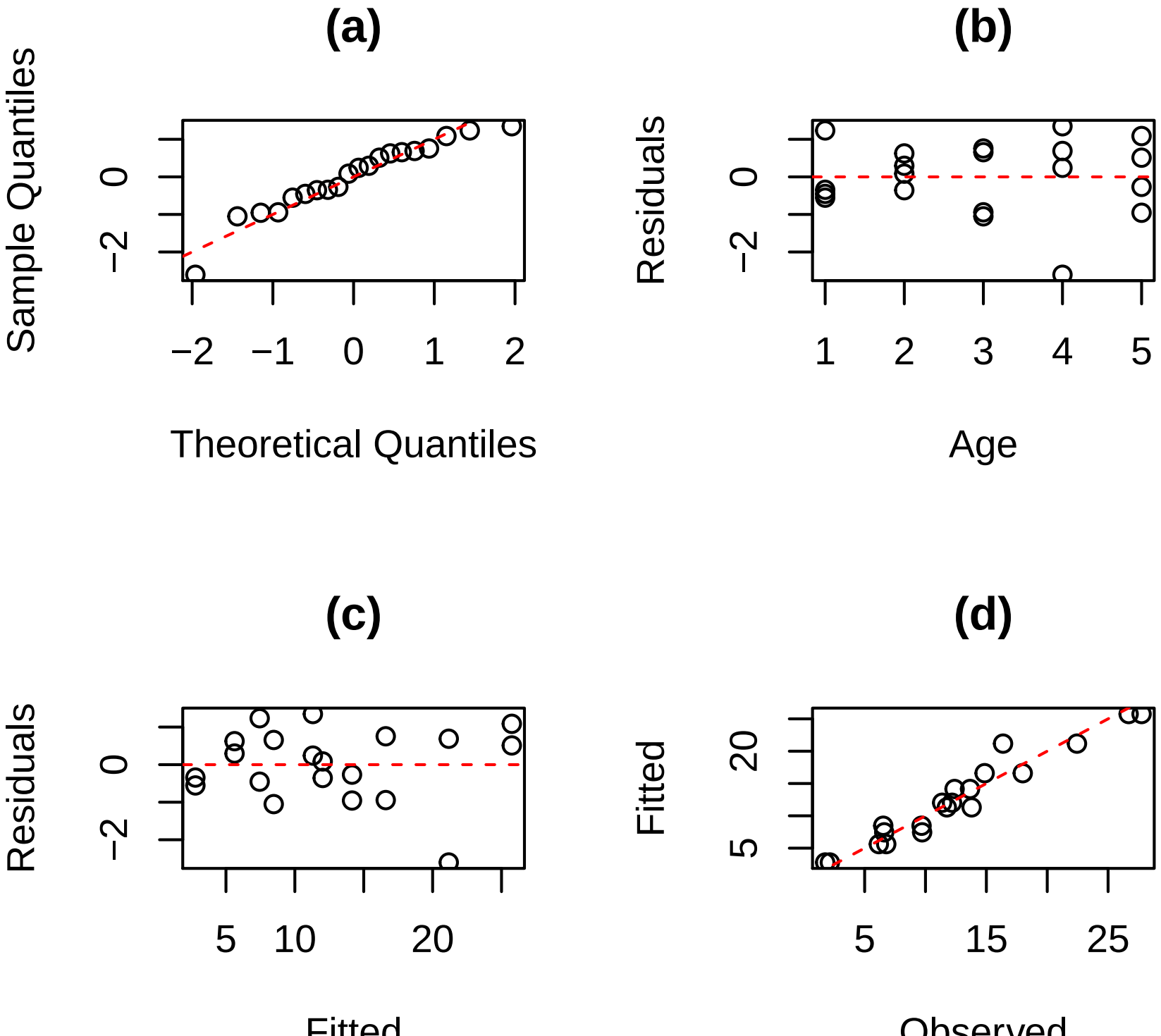
The code below forms residuals and then forms

- a QQ plot to assess normality,
- a plot of residuals versus age, to assess linearity,
- a plot of residuals versus fitted values, to see if an unmodeled mean-variance relationship) and
- a plot of fitted versus observed for an overall assessment of fit.

FTO: Residual analysis

```
residuals <- (liny - fitted)/sigmamed
par(mfrow = c(2, 2))
qqnorm(residuals, main = "")
title("(a)")
abline(0, 1, lty = 2, col = "red")
plot(residuals ~ linxa, ylab = "Residuals", xlab = "Age")
title("(b)")
abline(h = 0, lty = 2, col = "red")
plot(residuals ~ fitted, ylab = "Residuals", xlab = "Fitted")
title("(c)")
abline(h = 0, lty = 2, col = "red")
plot(fitted ~ liny, xlab = "Observed", ylab = "Fitted")
title("(d)")
abline(0, 1, lty = 2, col = "red")
```

The model assumptions do not appear to be greatly invalidated here.


(a)
Sample Quantiles
−2
0
−2
−1
0
1
2
Theoretical Quantiles
(b)
Residuals
−2
0
1
2
3
4
5
Age
(c)
Residuals
−2
0
5
10
20
Fitted
(d)
Fitted
5
20
5
15
25
Observed

Case-Control Example: INLA Analysis

We perform two analyses.

The first analysis uses the default priors in INLA (which are relatively flat).

```
x <- c(0, 1, 2)
y <- c(6, 8, 75)
z <- c(10, 66, 163)
cc.dat <- as.data.frame(rbind(y, z, x))
cc.mod <- inla(y ~ x, family = "binomial", data = cc.dat,
    Ntrials = y + z)
summary(cc.mod)
##
## Call:
## c("inla(formula = y ~ x, family = \"binomial\", data = cc.dat, Ntrials = y +
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          0.3496          0.1109          0.0293          0.4898
##
## Fixed effects:
##                mean     sd 0.025quant 0.5quant 0.975quant    mode kld
## (Intercept) -1.807 0.4554    -2.7495  -1.7901    -0.9594 -1.7556   0
## x            0.480 0.2505     0.0084   0.4726     0.9935  0.4574   0
##
## The model has no random effects
```

Case-Control Example: Stan Analysis

Analysis with default priors: uses code in file `LogiticExample.stan`

```
library(rstan)
stanlogist <- stan("LogisticExample.stan",
    data = list(x = c(0, 1, 2), y = c(6,
        8, 75), n = c(16, 74, 238)), iter = 1000,
    chains = 3, seed = 1234)
```

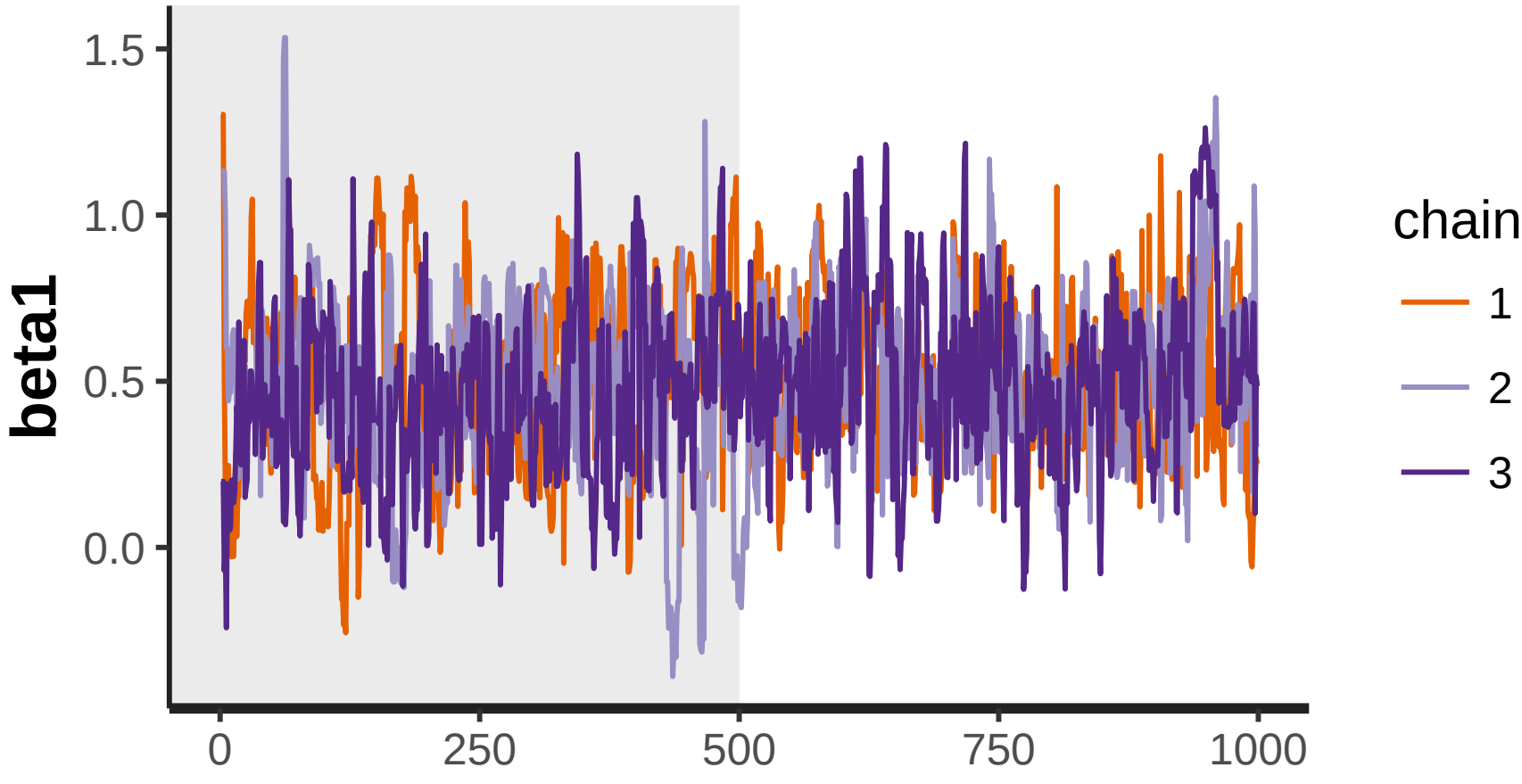
Case-Control Example: Stan Analysis

Close agreement with INLA analysis

```
summary(stanlogist)
## $summary
##              mean    se_mean        sd         2.5%          25%
## beta0   -1.896741 0.02864638 0.4448230   -2.8760657   -2.1946987
## beta1    0.521938 0.01530179 0.2416331    0.0798891    0.3575484
## lp__  -190.671902 0.05950391 0.9517759 -193.3467711 -191.0668758
##               50%          75%       97.5%    n_eff     Rhat
## beta0   -1.852174   -1.5872526   -1.096602 241.1210 1.002127
## beta1    0.510536    0.6805177    1.061702 249.3609 1.001741
## lp__  -190.339848 -189.9940112 -189.763856 255.8459 1.005647
##
## $c_summary
## , , chains = chain:1
##
##          stats
## parameter         mean        sd         2.5%          25%          50%
##     beta0   -1.8987981 0.4048038   -2.7063055   -2.2012985   -1.8680188
##     beta1    0.5214187 0.2217116    0.1295059    0.3566084    0.5102398
##     lp__  -190.5891262 0.8053920 -192.7789789 -190.9615323 -190.3446409
##          stats
## parameter          75%        97.5%
##     beta0   -1.5929847   -1.1913323
##     beta1    0.6814967    0.9621423
##     lp__  -190.0052048 -189.7605245
##
## , , chains = chain:2
##
##          stats
## parameter         mean        sd         2.5%          25%          50%
##     beta0   -1.8702653 0.4276670   -2.7634166   -2.1477799   -1.8297800
##     beta1    0.5086492 0.2308418    0.0896922    0.3555358    0.4866342
##     lp__  -190.6123879 0.9076313 -193.0716075 -190.9348057 -190.2923899
##          stats
```

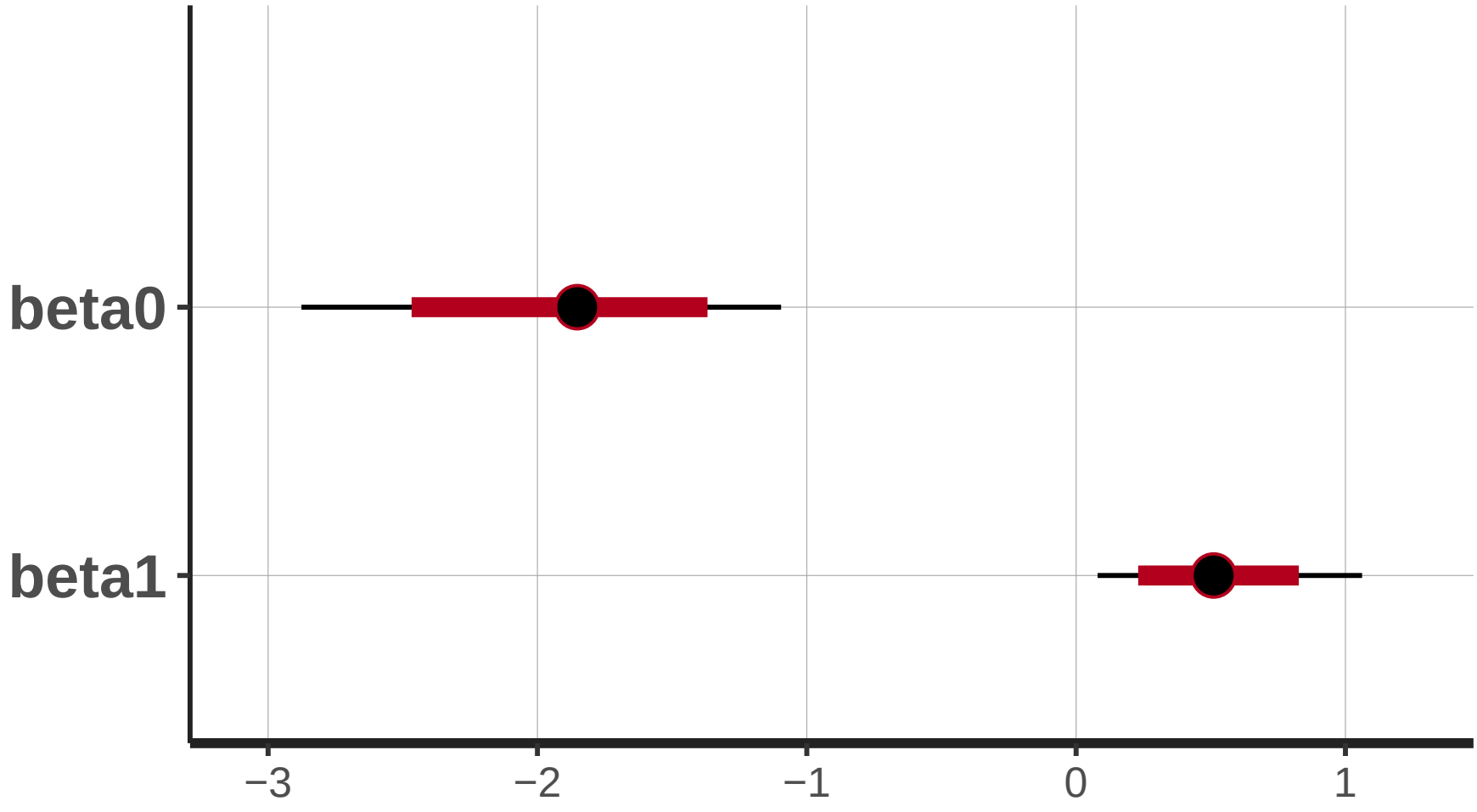
Case-Control Example: Stan Analysis

```
traceplot(stanlogist, pars = c("beta1"), inc_warmup = TRUE)
```



Case-Control Example: Stan Analysis

```
plot(stanlogist, color = "green")
```



Prior choice

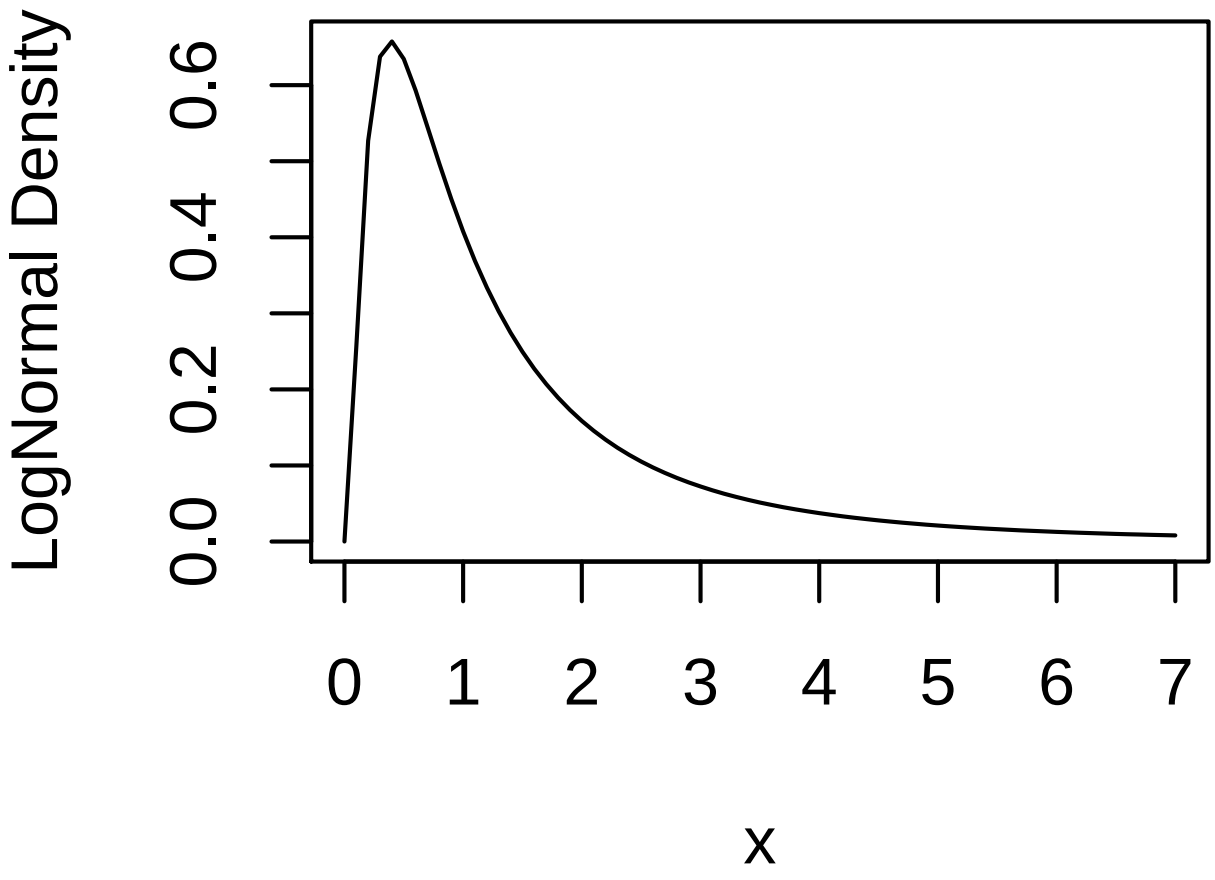
Suppose that for the odds ratio e^{β} we believe there is a 50% chance that the odds ratio is less than 1 and a 95% chance that it is less than 5; with $q_1 = 0.5, \theta_1 = 1.0$ and $q_2 = 0.95, \theta_2 = 5.0$, we obtain lognormal parameters $\mu = 0$ and $\sigma = (\log 5)/1.645 = 0.98$.

There is a function in the `SpatialEpi` package to find the parameters, as we illustrate.

```
library(SpatialEpi)
lnprior <- LogNormalPriorCh(1, 5, 0.5, 0.95)
lnprior
## $mu
## [1] 0
##
## $sigma
## [1] 0.9784688
```

Prior choice

```
plot(seq(0, 7, 0.1), dlnorm(seq(0, 7, 0.1), meanlog = lnprior$mu,
    sdlog = lnprior$sigma), type = "l", xlab = "x",
    ylab = "LogNormal Density")
```



Case-Control Example: Stan Analysis with Informative Prior

```
# Now with informative priors
W <- LogNormalPriorCh(1, 1.5, 0.5, 0.975)$sigma^2
cc.mod2 <- inla(y ~ x, family = "binomial", data = cc.dat,
    Ntrials = y + z, control.fixed = list(mean.intercept = c(0),
        prec.intercept = c(0.1), mean = c(0), prec = c(1/W)))
summary(cc.mod2)
##
## Call:
## c("inla(formula = y ~ x, family = \"binomial\", data = cc.dat, Ntrials = y +
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          0.3899          0.1269          0.0311          0.5478
##
## Fixed effects:
##                mean     sd 0.025quant 0.5quant 0.975quant    mode kld
## (Intercept) -1.3228 0.2896    -1.9011  -1.3194    -0.7633 -1.3127   0
## x            0.1987 0.1536    -0.1000   0.1977     0.5025  0.1958   0
##
## The model has no random effects
##
## The model has no hyperparameters
##
## Expected number of effective parameters(std dev): 1.441(0.00)
```

Case-Control Example: Stan Analysis with Informative Prior

```
library(rstan)
stanlogist2 <- stan("LogisticExamplePriors.stan",
    data = list(x = c(0, 1, 2), y = c(6,
        8, 75), n = c(16, 74, 238)), iter = 1000,
    chains = 3, seed = 2345)
```

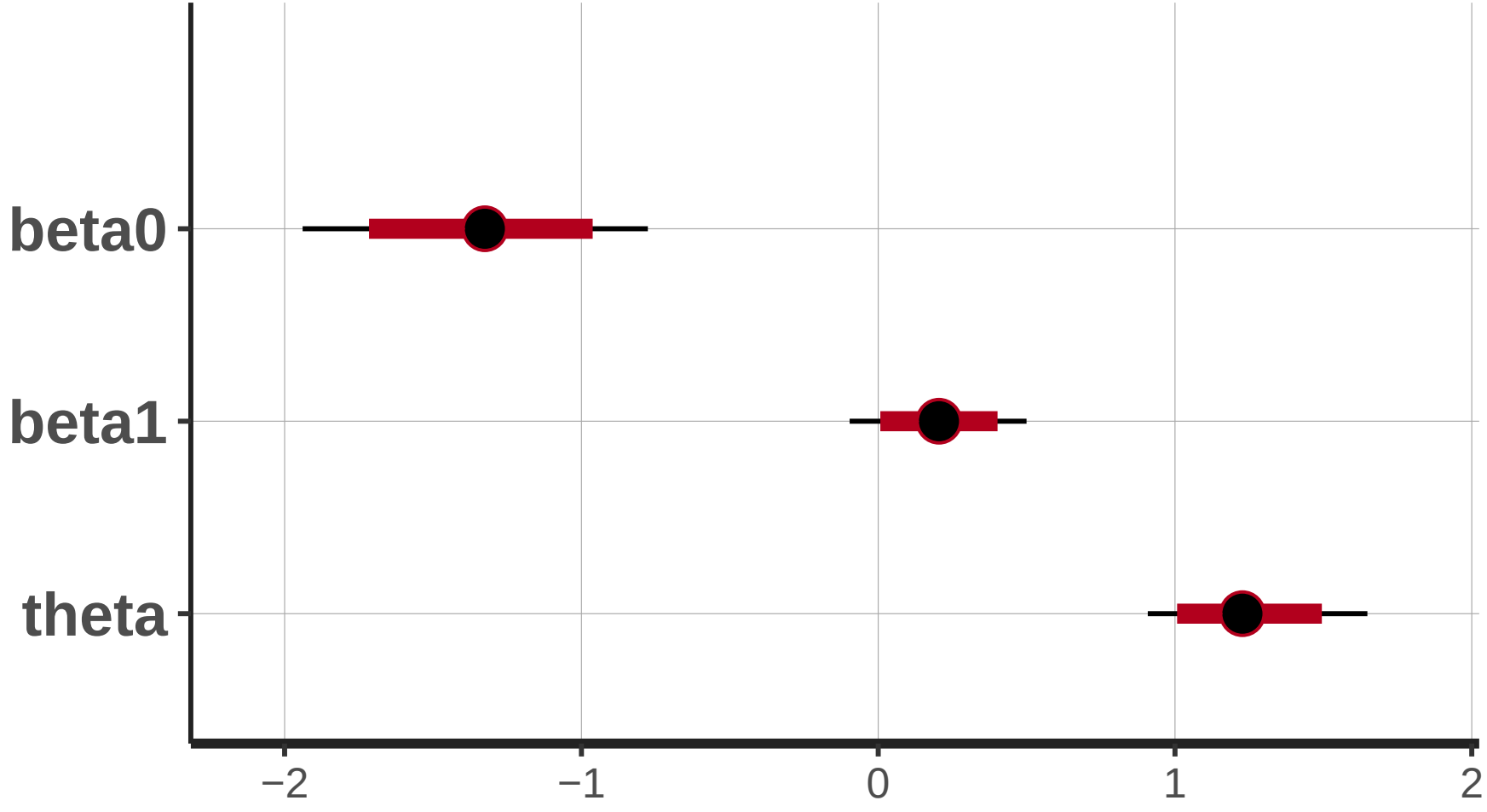
Case-Control Example: Stan Analysis with Informative Prior

Again close agreement with INLA analysis

```
summary(stanlogist2)
## $summary
##                mean     se_mean         sd           2.5%           25%
## beta0   -1.3320620 0.014269441 0.2949263   -1.93926015   -1.52449000
## beta1    0.2024719 0.007232670 0.1531556   -0.09614039    0.09922433
## theta    1.2388350 0.008906524 0.1902865    0.90833674    1.10431404
## lp__  -191.9624650 0.045033911 0.9980523 -194.82761315 -192.34112928
##                50%          75%        97.5%    n_eff     Rhat
## beta0   -1.3251986   -1.1415130   -0.7752082 427.1821 1.015965
## beta1    0.2046701    0.3007626    0.4996593 448.4030 1.015475
## theta    1.2271202    1.3508891    1.6481598 456.4565 1.014184
## lp__  -191.6608107 -191.2507037 -190.9755725 491.1649 1.000064
##
## $c_summary
## , , chains = chain:1
##
##          stats
## parameter         mean        sd          2.5%           25%          50%
##     beta0   -1.3314130 0.2948618   -1.93490687   -1.51810415   -1.3231435
##     beta1    0.2008541 0.1504716   -0.08069469    0.09573295    0.2009142
##     theta    1.2363721 0.1876948    0.92247684    1.10046525    1.2225205
##     lp__  -191.9470795 0.9680294 -194.53120586 -192.30144345 -191.6602020
##          stats
## parameter          75%        97.5%
##     beta0   -1.1438971   -0.7358760
##     beta1    0.2968426    0.5023257
##     theta    1.3456038    1.6525672
##     lp__  -191.2548581 -190.9747250
##
## , , chains = chain:2
```

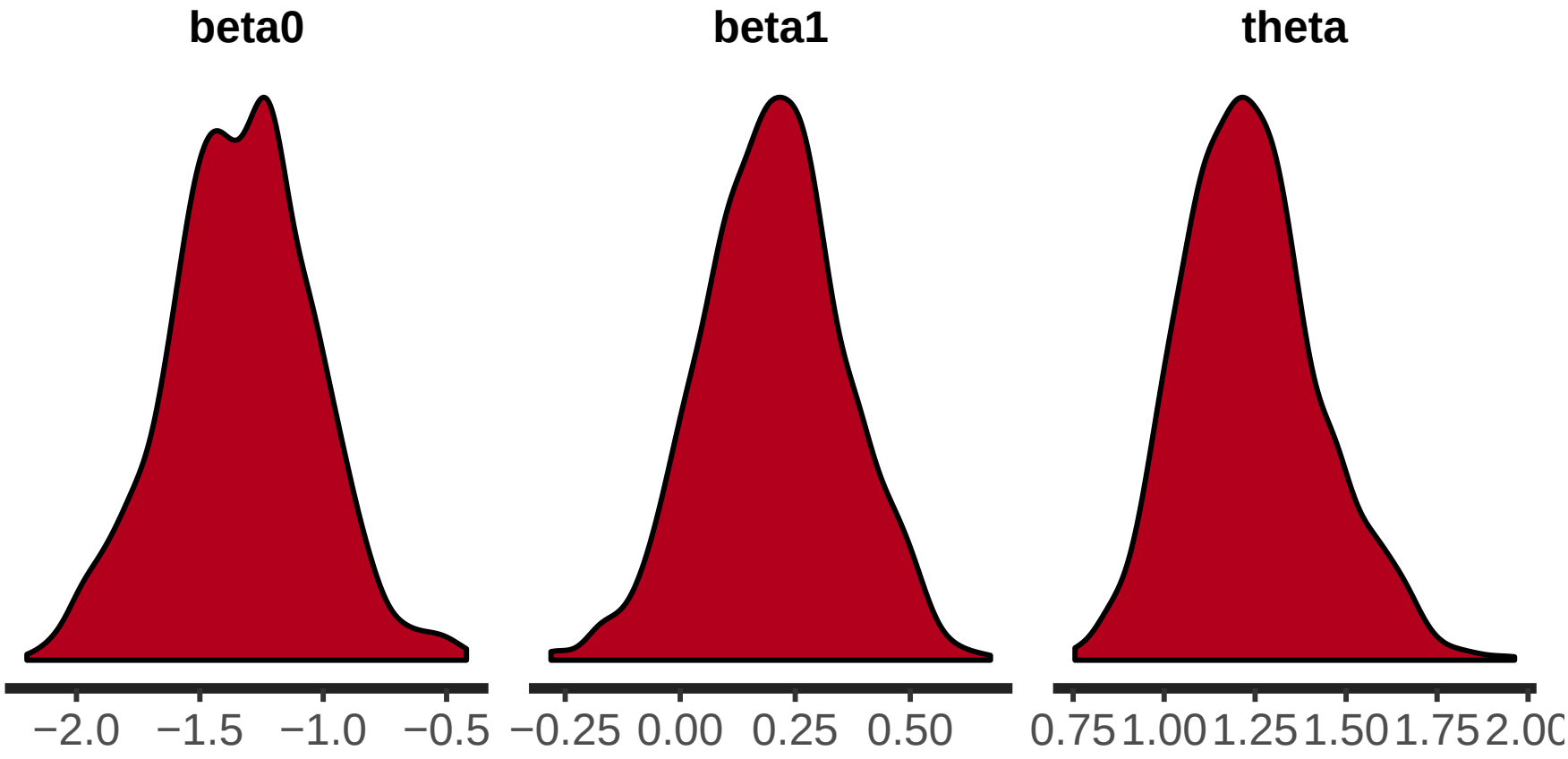
Case-Control Example: Stan Analysis with Informative Prior

```
plot(stanlogist2, color = "green", parameter = "theta")
```



Case-Control Example: Stan Analysis with Informative Prior

```
stan_dens(stanlogist2)
```



A Simple ANOVA Example

We begin with simulated data from the simple one-way ANOVA model example:

$$\begin{aligned} Y_{ij}|\beta_0, b_i &= \beta_0 + b_i + \epsilon_{ij} \\ \epsilon_{ij}|\sigma^2_\epsilon &\sim_{iid} \text{normal}(0, \sigma^2_\epsilon) \\ b_i|\sigma^2_b &\sim_{iid} \text{normal}(0, \sigma^2_b) \end{aligned}$$

$i = 1, ..., 10; j = 1, ..., 5$, with $\beta_0 = 0.5$, $\sigma^2_\epsilon = 0.2^2$ and $\sigma^2_b = 0.3^2$.

b_i are random effects and ϵ_{ij} are measurement errors and there are two variances to estimate, σ^2_ϵ and σ^2_b.

In a fixed effects Bayesian model, the variance σ^2_b would be fixed in advance.

A Simple ANOVA Example

Simulation is described and analyzed below. We fit the one-way ANOVA model and see reasonable recovery of the true values that were used to simulate the data.

Not a big surprise, since we have fitted the model that was used to simulate the data!

```
set.seed(133)
sigma.b <- 0.3
sigma.e <- 0.2
m <- 10
ni <- 5
beta0 <- 0.5
b <- rnorm(m, mean = 0, sd = sigma.b)
e <- rnorm(m * ni, mean = 0, sd = sigma.e)
Yvec <- beta0 + rep(b, each = ni) + e
simdata <- data.frame(y = Yvec, ind = rep(1:m, each = ni))
result <- inla(y ~ f(ind, model = "iid"), data = simdata)
sigma.est <- 1/sqrt(result$summary.hyperpar[, 4])
sigma.est
## [1] 0.1927621 0.2455393
```

`sigma.est` corresponds to the posterior medians of σ_ϵ and σ_b, respectively.

A Simple ANOVA Example

```
summary(result)
##
## Call:
## "inla(formula = y ~ f(ind, model = \"iid\"), data = simdata)"
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          0.4745          0.2046          0.0521          0.7312
##
## Fixed effects:
##               mean     sd 0.025quant 0.5quant 0.975quant  mode kld
## (Intercept) 0.525 0.0881      0.349    0.525     0.7008 0.525   0
##
## Random effects:
## Name   Model
##  ind   IID model
##
## Model hyperparameters:
##                                              mean    sd 0.025quant 0.5quant
## Precision for the Gaussian observations 27.42 6.026     17.184    26.91
## Precision for ind                       18.45 9.368      6.048    16.59
##                                         0.975quant  mode
## Precision for the Gaussian observations      40.72 25.98
## Precision for ind                            41.78 13.15
##
## Expected number of effective parameters(std dev): 8.896(0.5733)
```

RNA Seq Analysis

We fit a hierarchical logistic regression model starting with first stage:

$$Y_{ij}|N_{ij}, p_{ij} \sim \text{binomial}(N_{ij}, p_{ij})$$

so that p_{ij} is the probability of seeing an A read for gene i and replicate j, $i = 1, \ldots, 10$, $j = 1, 2$.

Then the odds of an A read is $\frac{p_{ij}}{1-p_{ij}}$.

At the second stage:

$$\text{logit } p_{ij} \quad = \quad \theta_i + \epsilon_{ij}$$

where $\epsilon_{ij}|\sigma^2 \sim \text{normal}(0, \sigma^2)$ represent random effects that allow for excess-binomial variation; there are a pair for each gene.

The θ_i parameters are taken as fixed effects with a relatively flat prior (the default choice in `INLA`).

$\exp(\theta_i)$ is the odds of seeing an A read for gene i.

RNA Seq Analysis

Rows 1 and 2 represent the two replicates for gene 1, rows 3 and 4 represent the two replicates for gene 2, etc. . .

`rep1` is the variable that defines the random effects.

`xvar` is the gene number, there are 10 genes in this dataset.

```
rnay <- c(1963, 3676, 249, 110, 78, 92, 1585, 798,
    2525, 2620, 598, 473, 120, 72, 1496, 1291, 397,
    480, 242, 174)
rnan <- c(7617, 10413, 308, 114, 161, 153, 2321, 1527,
    4142, 3861, 910, 778, 160, 85, 2795, 2697, 810,
    928, 466, 313)
rnarep1 <- seq(1, 20)
rnaxvar <- rep(1:10, each = 2)
RNAdat <- as.data.frame(cbind(rnay, rnan, rnarep1,
    rnaxvar))
names(RNAdat) <- c("y", "n", "rep1", "xvar")
head(RNAdat)
##      y     n rep1 xvar
## 1 1963  7617    1    1
## 2 3676 10413    2    1
## 3  249   308    3    2
## 4  110   114    4    2
## 5   78   161    5    3
## 6   92   153    6    3
```

RNA Seq Analysis

Below is the code for fitting the random effects model.

The -1 in the model specification removes the intercept, so that the factor levels are defined with one level for each gene.

```
RNAfit <- inla(y ~ as.factor(xvar) - 1 + f(rep1, model = "iid"),
    family = "binomial", data = RNAdat, Ntrials = n)
```

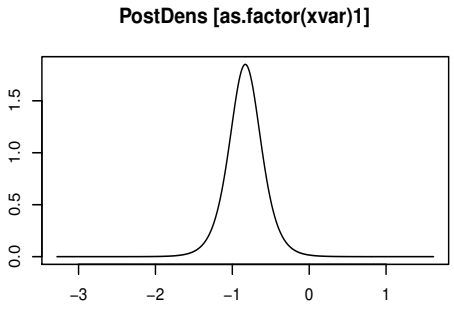
RNA Seq Analysis

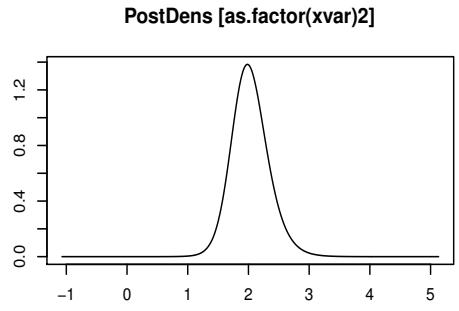
```
RNAfit$summary.fixed[1:10, c(1:5)]
##                             mean         sd  0.025quant    0.5quant 0.975quant
## as.factor(xvar)1  -0.83155399 0.2449397 -1.32495899 -0.83151685 -0.3382479
## as.factor(xvar)2   2.03068064 0.3101873  1.46559099  2.01343791  2.6956807
## as.factor(xvar)3   0.17300739 0.2695801 -0.36421060  0.17273125  0.7112491
## as.factor(xvar)4   0.43017910 0.2467214 -0.06671066  0.43035859  0.9262246
## as.factor(xvar)5   0.59643644 0.2454980  0.10218410  0.59640667  1.0908708
## as.factor(xvar)6   0.54559183 0.2495715  0.04397161  0.54561709  1.0470165
## as.factor(xvar)7   1.36464386 0.2953853  0.79058757  1.36028332  1.9636085
## as.factor(xvar)8   0.02797282 0.2458656 -0.46697974  0.02796678  0.5229928
## as.factor(xvar)9   0.01493824 0.2490628 -0.48574616  0.01495393  0.5154998
## as.factor(xvar)10  0.14969702 0.2551034 -0.36125143  0.14947307  0.6616085
RNAfit$summary.hyperpar[c(1:5)]
##                           mean       sd 0.025quant 0.5quant 0.975quant
## Precision for rep1 11.70761 6.656763   3.130449 10.26053   28.61355
```

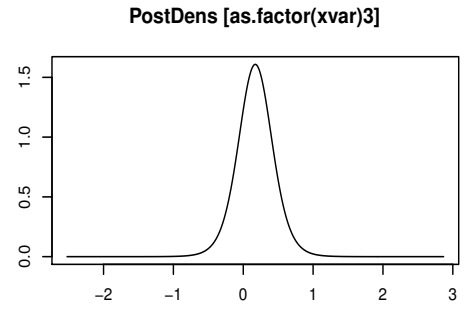
RNA Seq Analysis

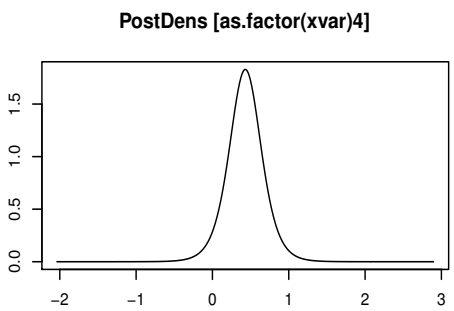
```
RNAfit$summary.random$rep1[, c(2:6)]
##            mean        sd 0.025quant    0.5quant 0.975quant
## 1  -0.22468096 0.2449378 -0.7185219 -0.22444808 0.26813090
## 2   0.22458160 0.2449376 -0.2682184  0.22434222 0.71855071
## 3  -0.47920734 0.3075406 -1.1588450 -0.45392652 0.05666759
## 4   0.47945549 0.3076087 -0.0574999  0.45407211 1.15883325
## 5  -0.18505145 0.2651825 -0.7295237 -0.17963840 0.33106737
## 6   0.18507216 0.2651837 -0.3309513  0.17966356 0.73001122
## 7   0.32965005 0.2467111 -0.1636680  0.32828431 0.82950375
## 8  -0.32959873 0.2467106 -0.8290262 -0.32826069 0.16345690
## 9  -0.14888308 0.2454863 -0.6438145 -0.14859542 0.34475554
## 10  0.14895430 0.2454864 -0.3446255  0.14863331 0.64409625
## 11  0.10009659 0.2493055 -0.3990493  0.09923562 0.60342615
## 12 -0.10003145 0.2493052 -0.6031011 -0.09920138 0.39892846
## 13 -0.19209500 0.2818424 -0.7809885 -0.18255186 0.34481092
## 14  0.19225903 0.2818599 -0.3440713  0.18273891 0.78073095
## 15  0.11140071 0.2458431 -0.3828394  0.11109097 0.60714155
## 16 -0.11139740 0.2458431 -0.6069693 -0.11111024 0.38286208
## 17 -0.05155325 0.2488329 -0.5527594 -0.05116570 0.44773131
## 18  0.05155501 0.2488328 -0.4478111  0.05114553 0.55288798
## 19 -0.06583790 0.2540326 -0.5781502 -0.06481665 0.44113750
## 20  0.06585581 0.2540332 -0.4414835  0.06481154 0.57837890
```

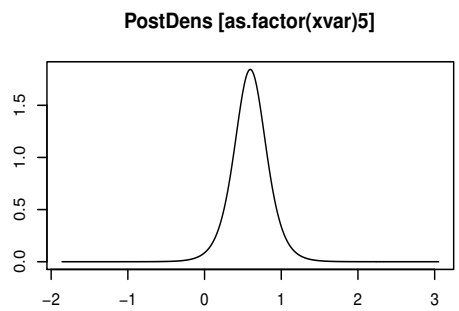
RNA Seq Analysis

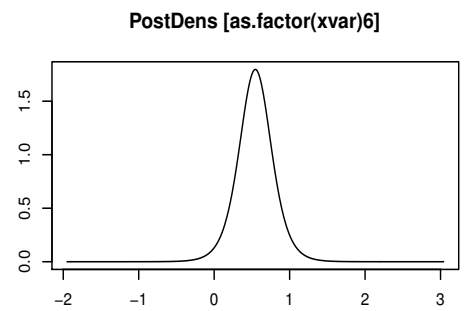


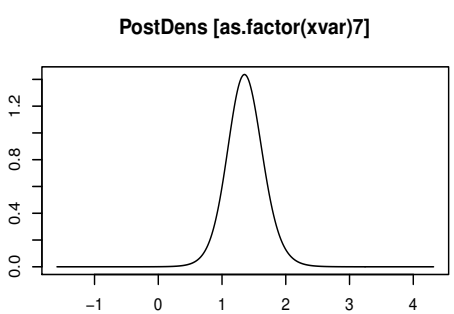


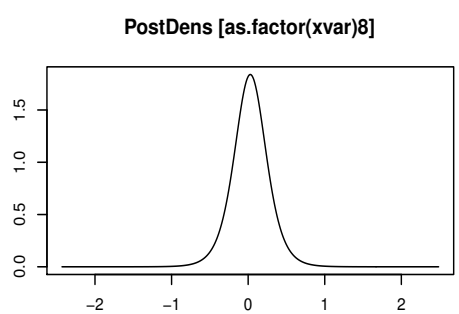


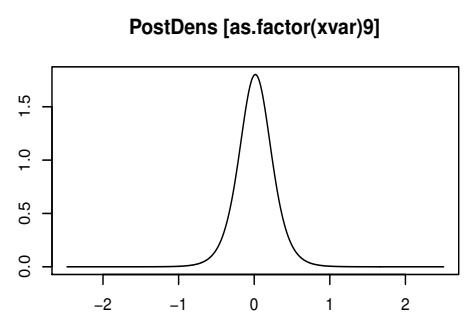




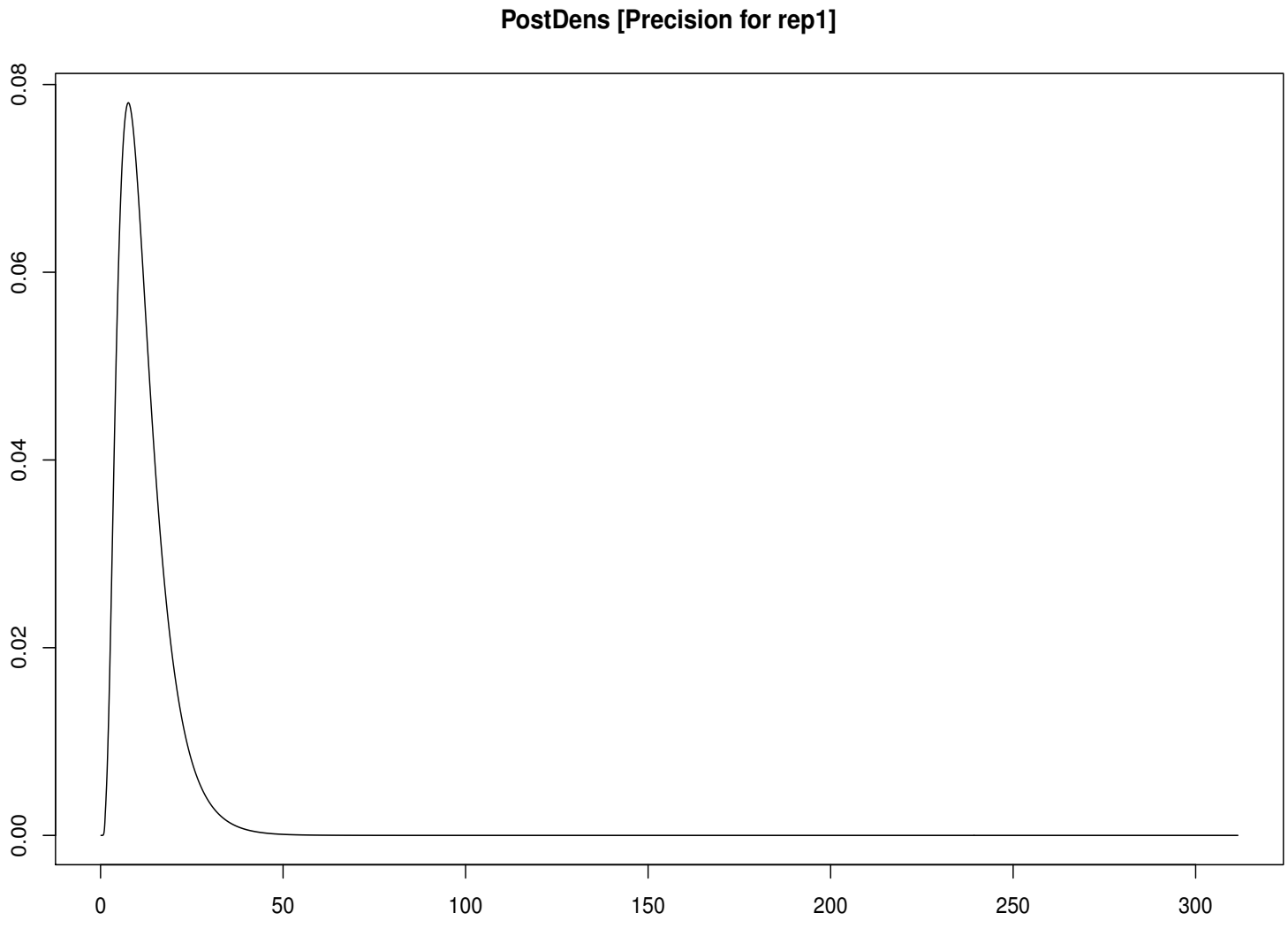




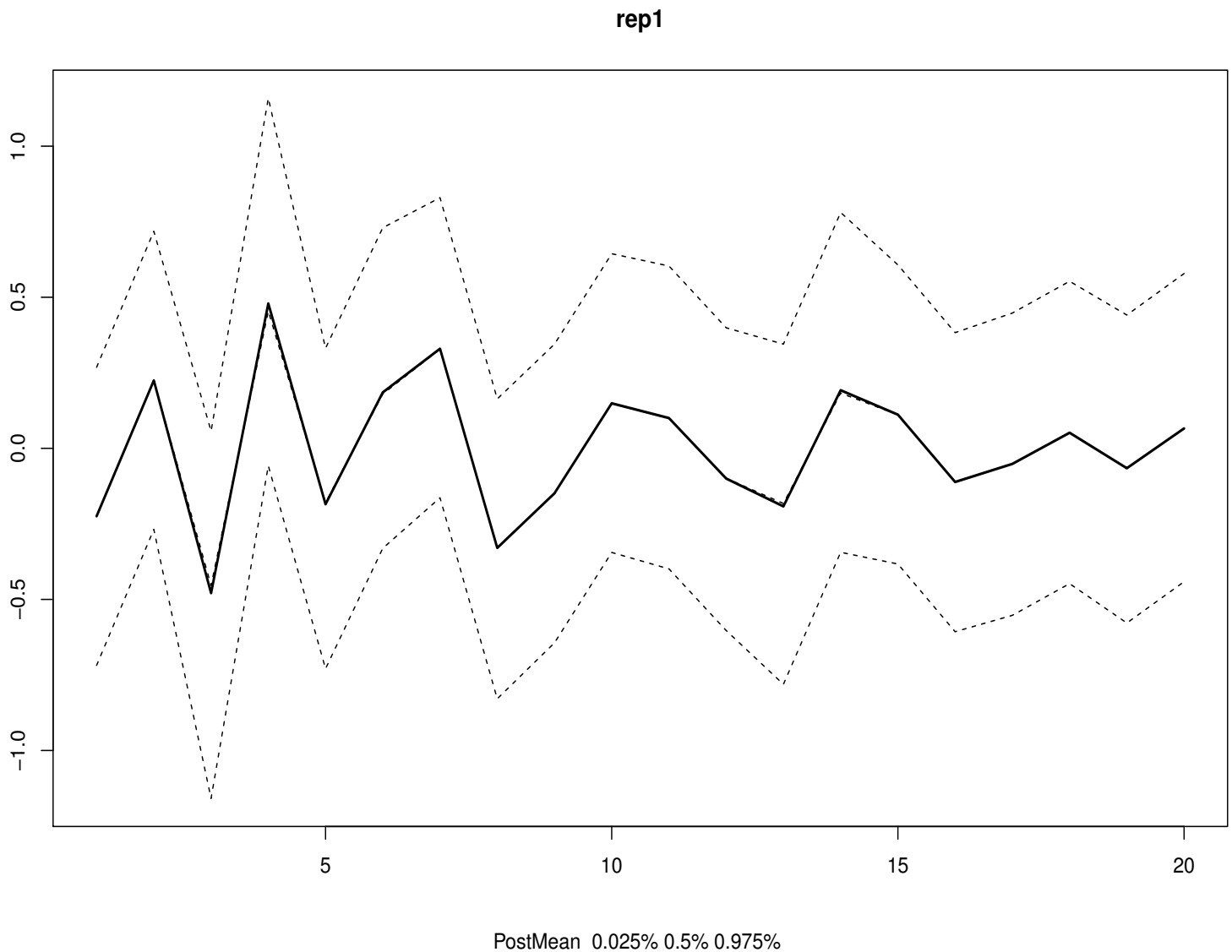




RNA Seq Analysis



RNA Seq Analysis



RNA Seq Analysis

We extract the 95% intervals and posterior medians for the log odds of being an A allele.

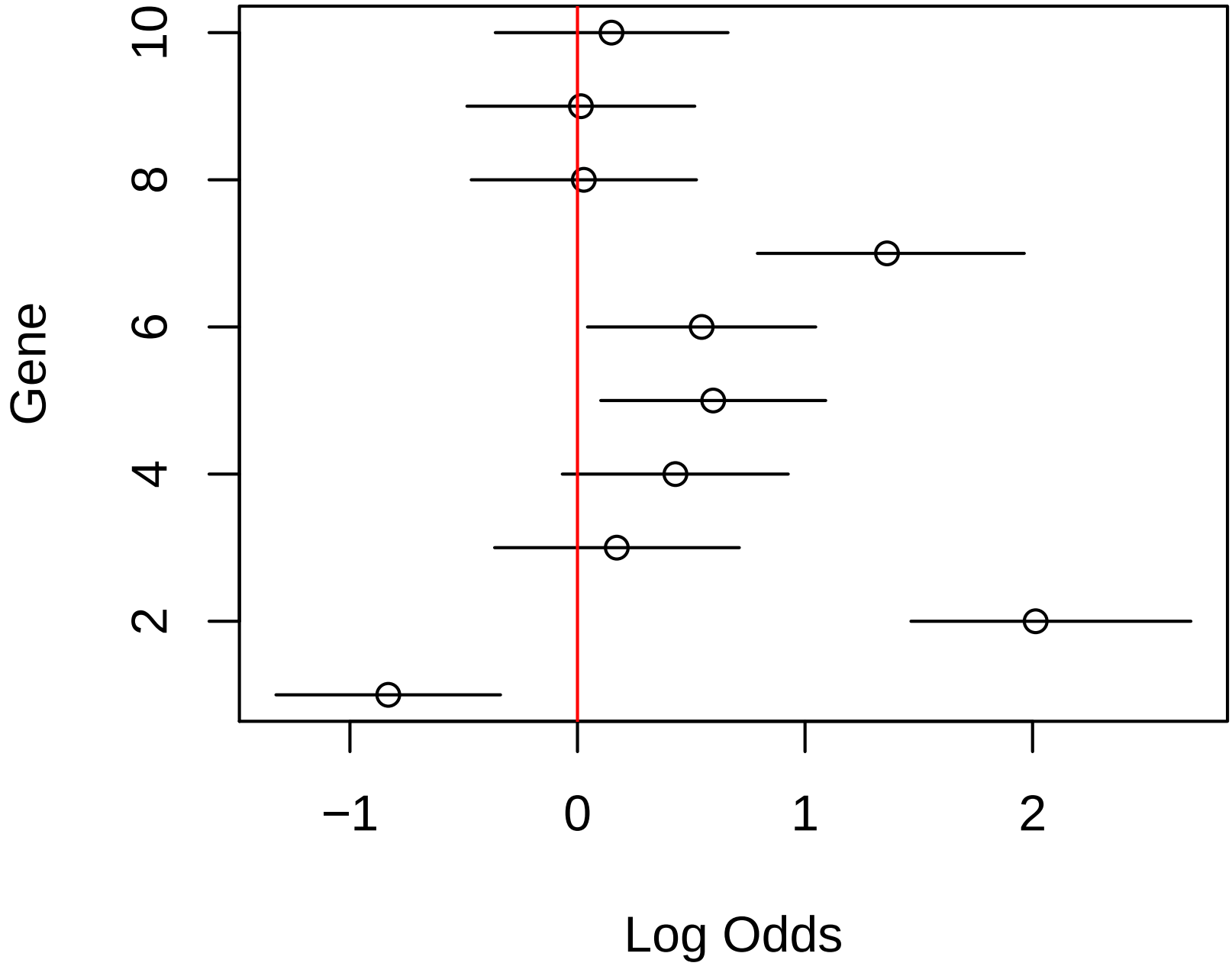
Comparison with 0 gives an indication of cis effects.

Genes 1, 2, 5, 6, 7 show evidence of cis effects.

RNA Seq Analysis

```
thetasum <- RNAfit$summary.fixed[, 3:5]
par(mfrow = c(1, 1))
plot(thetasum[, 2], seq(1, 10), xlim = c(min(thetasum),
    max(thetasum)), ylab = "Gene", xlab = "Log Odds")
for (i in 1:10) {
    lines(x = c(thetasum[i, 1], thetasum[i, 3]), y = c(i,
        i))
}
abline(v = 0, col = "red")
# Intervals to the left/right of this line?
```

RNA Seq Analysis



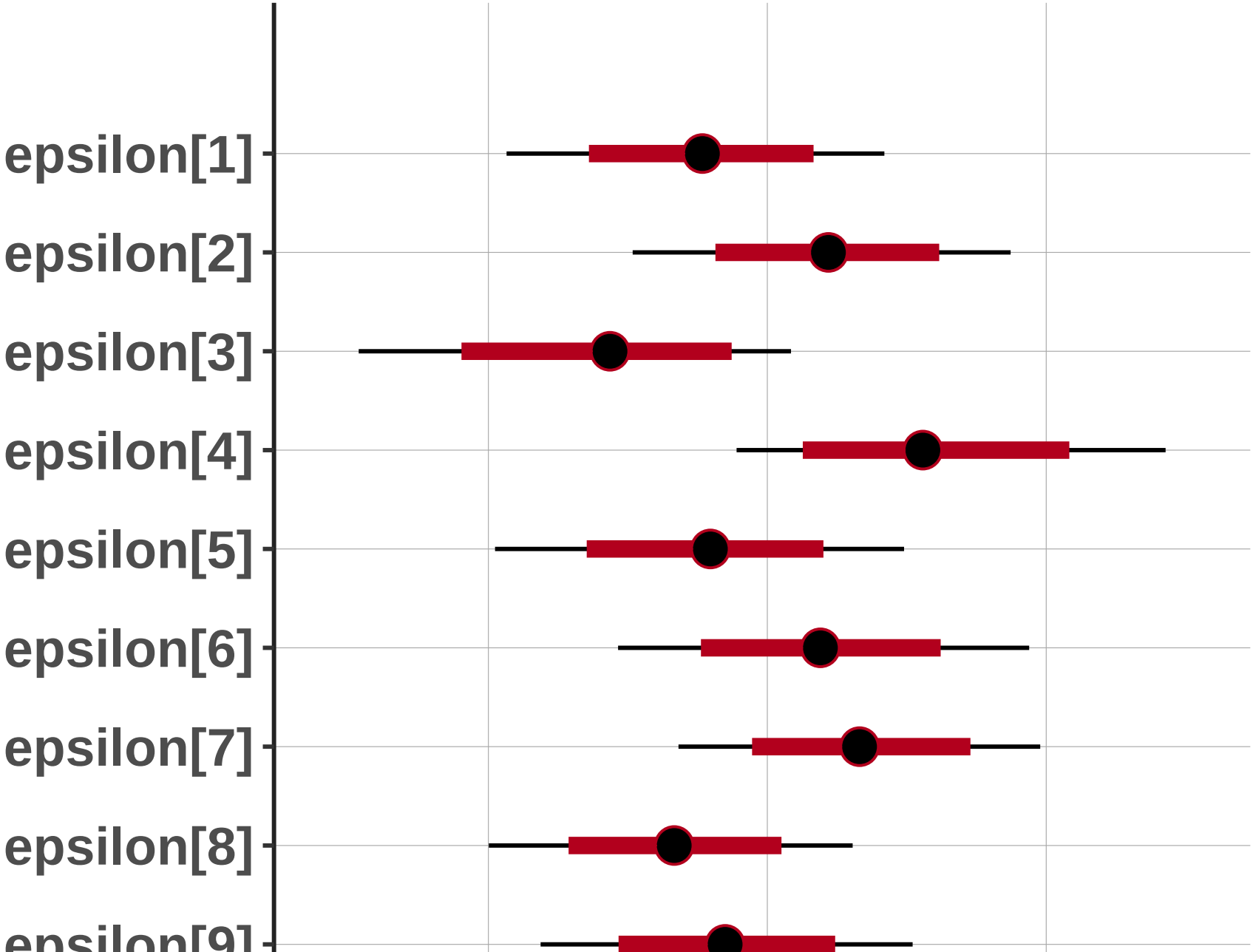
Logistic regression with random effects: Stan analysis

```
N <- 20
rnay <- c(1963, 3676, 249, 110, 78, 92, 1585,
    798, 2525, 2620, 598, 473, 120, 72, 1496,
    1291, 397, 480, 242, 174)
rnan <- c(7617, 10413, 308, 114, 161, 153,
    2321, 1527, 4142, 3861, 910, 778, 160,
    85, 2795, 2697, 810, 928, 466, 313)
rnax <- c(1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6,
    6, 7, 7, 8, 8, 9, 9, 10, 10)
rnadat3 <- list(N = N, rnay = rnay, rnan = rnan,
    rnax = rnax)
stanlogist3 <- stan("LogisticExampleRandomEffects.stan",
    data = rnadat3, iter = 3000, chains = 3,
    seed = 3456)
```

Logistic regression with random effects: Stan analysis

```
## $summary
##                       mean     se_mean        sd          2.5%
## epsilon[1]  -2.352394e-01 0.005035523 0.3377932 -9.348043e-01
## epsilon[2]   2.149715e-01 0.005025605 0.3371278 -4.825299e-01
## epsilon[3]  -5.937627e-01 0.005914763 0.3967744 -1.464918e+00
## epsilon[4]   5.838040e-01 0.005812034 0.3898831 -1.099901e-01
## epsilon[5]  -2.158195e-01 0.005387864 0.3614289 -9.761000e-01
## epsilon[6]   1.933069e-01 0.005387529 0.3614064 -5.348897e-01
## epsilon[7]   3.319339e-01 0.004939511 0.3313525 -3.184918e-01
## epsilon[8]  -3.356738e-01 0.005006729 0.3311913 -9.983966e-01
## epsilon[9]  -1.478749e-01 0.004939808 0.3313724 -8.128215e-01
## epsilon[10]  1.511476e-01 0.004921684 0.3301566 -5.054824e-01
## epsilon[11]  1.068929e-01 0.005135464 0.3444974 -5.991849e-01
## epsilon[12] -9.834349e-02 0.005147382 0.3452969 -8.049331e-01
## epsilon[13] -2.249514e-01 0.005542030 0.3717707 -1.000760e+00
## epsilon[14]  2.267120e-01 0.005542069 0.3717733 -4.938545e-01
## epsilon[15]  1.100893e-01 0.004879527 0.3273286 -5.697033e-01
## epsilon[16] -1.145527e-01 0.004862350 0.3261763 -7.823653e-01
## epsilon[17] -5.820523e-02 0.005190256 0.3481730 -7.718849e-01
## epsilon[18]  4.668302e-02 0.005184264 0.3477710 -6.602767e-01
## epsilon[19] -6.959867e-02 0.005006266 0.3358305 -7.581060e-01
## epsilon[20]  6.555272e-02 0.005052575 0.3389370 -6.374482e-01
## sigma        4.418880e-01 0.005213518 0.1544342  2.334197e-01
## beta[1]     -8.213793e-01 0.005025943 0.3371505 -1.464696e+00
## beta[2]      2.120010e+00 0.005905927 0.3961817  1.400239e+00
## beta[3]      1.900119e-01 0.005439580 0.3648981 -5.312682e-01
```

Logistic regression with random effects: Stan analysis



Approximate Bayes

We return to the case control example seen earlier.

Below we construct the posterior by hand

```
x <- c(0, 1, 2)
y <- c(6, 8, 75)
z <- c(10, 66, 163)
logitmod <- glm(cbind(y, z) ~ x, family = "binomial")
thetahat <- logitmod$coef[2]
V <- vcov(logitmod)[2, 2]
# 97.5 point of prior is log(1.5) so that we with
# prob 0.95 we think theta lies in (2/3,1.5)
W <- LogNormalPriorCh(1, 1.5, 0.5, 0.975)$sigma^2
```

Approximate Bayes: estimation

```
r <- W/(V + W)
r
## [1] 0.4055539
# Not so much data here, so weight on prior is
# high.  Bayesian posterior median
exp(r * thetahat)
##        x
## 1.214286
# Shrunk towards prior median of 1 Note: INLA
# estimate (with same prior) is 1.22 and
# approximate posterior SD here is sqrt(rV)=0.159,
# INLA version is 0.154.  Bayesian approximate 95%
# credible interval
exp(r * thetahat - 1.96 * sqrt(r * V))
##         x
## 0.8882832
exp(r * thetahat + 1.96 * sqrt(r * V))
##        x
## 1.659932
```

Approximate Bayes: hypothesis testing

Now we turn to testing using Bayes factors.

We examine the sensitivity to the prior on the alternative, π_1.

```
pi1 <- c(1/2, 1/100, 1/1000, 1/10000, 1/1e+05)  # 5 prior probs on the null
source("http://faculty.washington.edu/jonno/BFDP.R")
BFcall <- BFDPfunV(thetahat, V, W, pi1)
BFcall
## $BF
##         x
## 0.6182773
##
## $pH0
##        x
## 0.256323
##
## $pH1
##         x
## 0.4145761
##
## $BFDP
## [1] 0.3820589 0.9839253 0.9983836 0.9998383 0.9999838
```

So data are twice as likely under the alternative (0.502) as compared to the null (0.256).