2015 SISG Bayesian Statistics for Genetics
R Notes: Generalized Linear Modeling

Jon Wakefield
Departments of Statistics and Biostatistics, University of Washington

2016-07-26

Case control example

We analyze some case control data using logistic regression models, first using likelihood methods.

```
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
```

Case control example

We fit the logistic regression model as a generalized linear model and then examine the estimate and an asymptotic 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

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

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

```
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

```
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.

```
# 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

```
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"
```

FTO example via INLA

The posterior means and 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) -0.06210829 1.3722780 -2.7761257 -0.06236768   2.648733
## linxg        2.93410215 1.9386548 -0.9019926  2.93442177   6.761854
## linxa        2.84250685 0.4138197  2.0236998  2.84255420   3.659648
## linxint      1.73238963 0.5847175  0.5760223  1.73226830   2.887467
##                    mode          kld
## (Intercept) -0.06271412 8.030481e-12
## linxg        2.93516637 7.842042e-12
## linxa        2.84267554 8.046944e-12
## linxint      1.73210118 7.900298e-12
```

FTO example via INLA

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

```
summary(lin.mod)
##
## Call:
## "inla(formula = formula, family = \"gaussian\", data = ftodf)"
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          0.9418          0.0786          0.0857          1.1061
##
## Fixed effects:
##                mean     sd 0.025quant 0.5quant 0.975quant    mode kld
## (Intercept) -0.0621 1.3723    -2.7761  -0.0624     2.6487 -0.0627   0
## linxg        2.9341 1.9387    -0.9020   2.9344     6.7619  2.9352   0
## linxa        2.8425 0.4138     2.0237   2.8426     3.6596  2.8427   0
## linxint      1.7324 0.5847     0.5760   1.7323     2.8875  1.7321   0
##
## The model has no random effects
##
## Model hyperparameters:
##                                                mean     sd 0.025quant 0.5quant
## Precision for the Gaussian observations 0.3047 0.1016     0.1455    0.292
##                                                0.975quant   mode
## Precision for the Gaussian observations     0.5391 0.2678
```

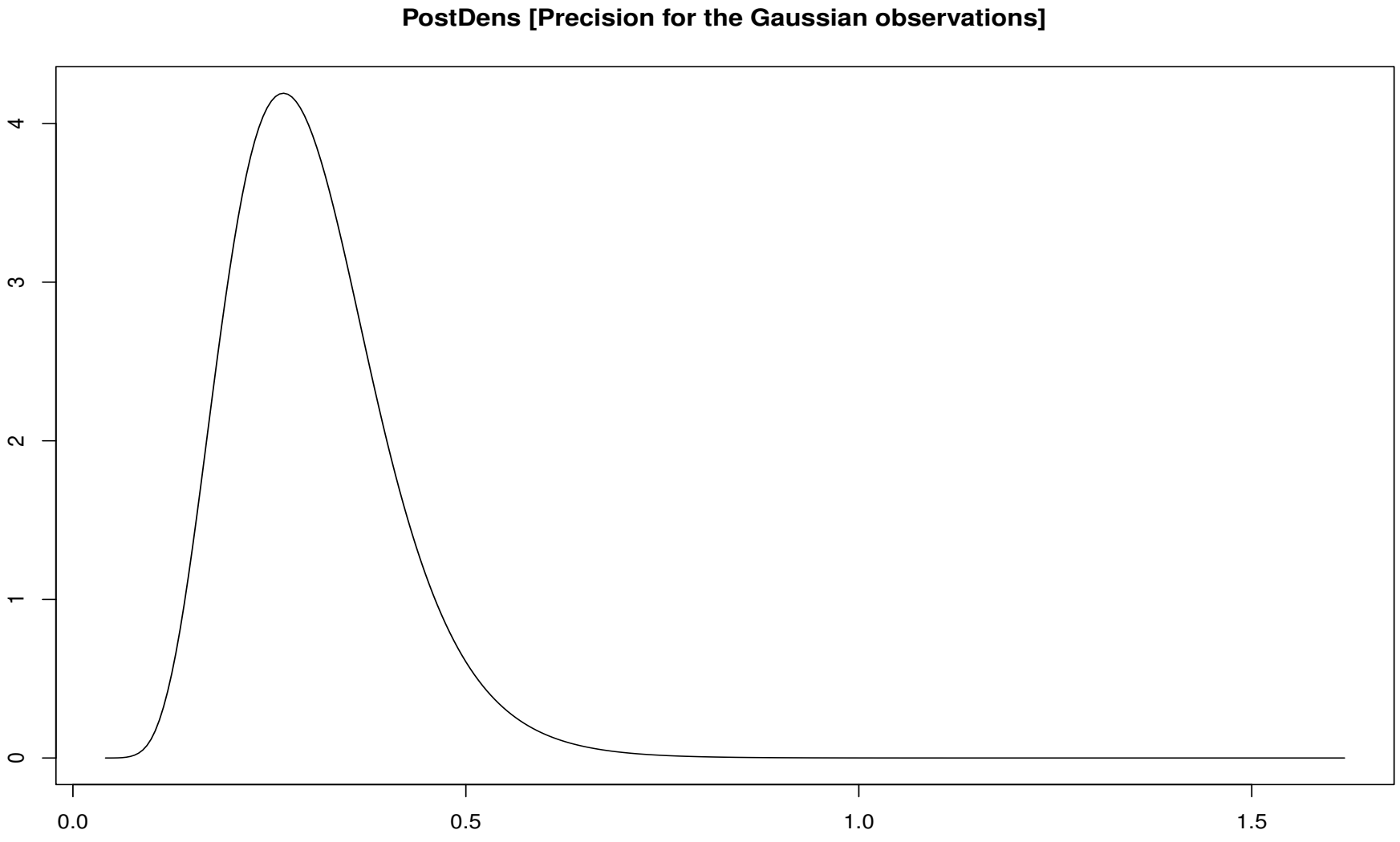
FTO Posterior marginals

We now examine the posterior marginal distributions.

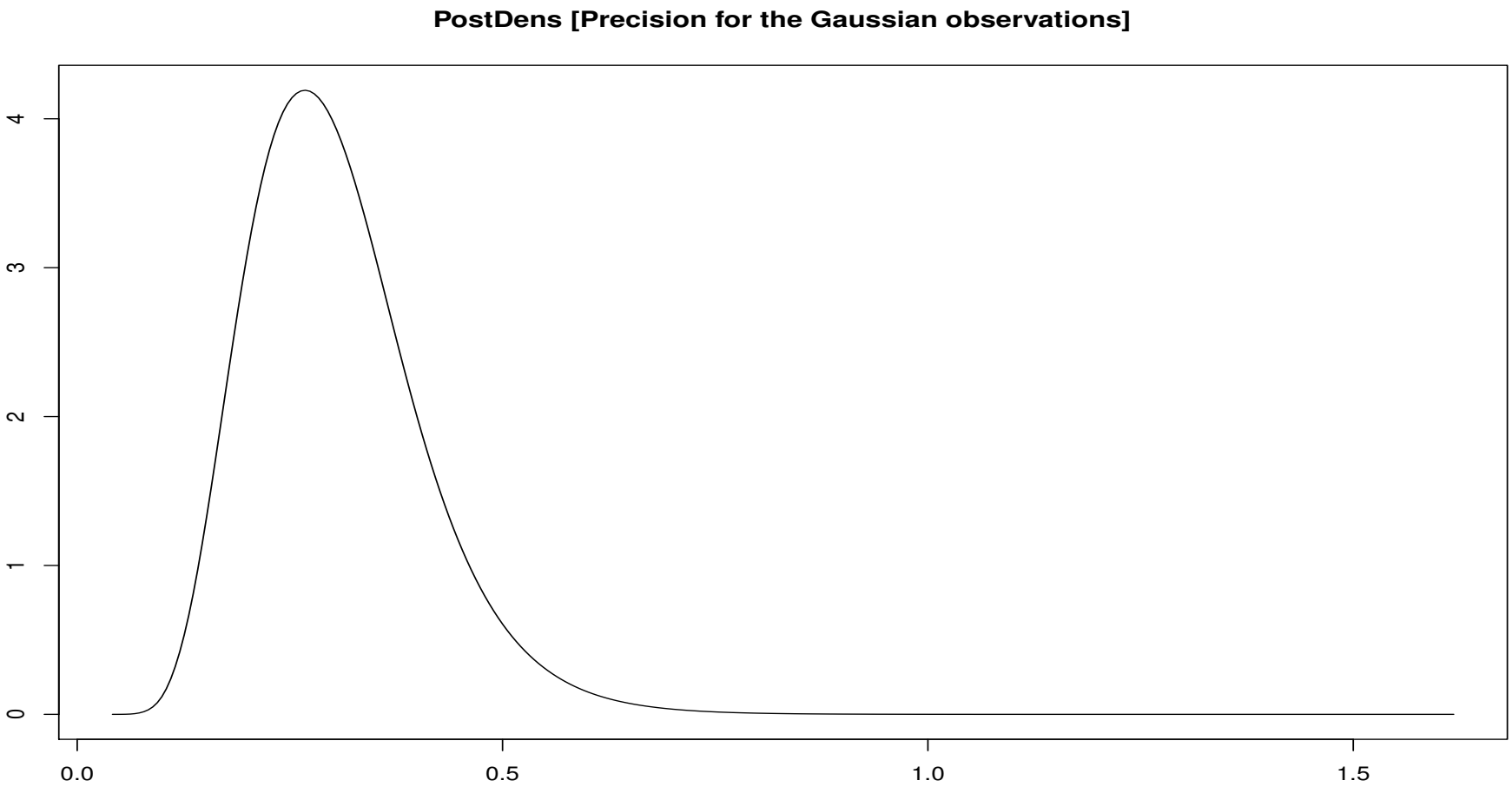
The posterior marginal distribution for the vector of regression coefficients (including the intercept) is given below, and then we examine the posterior marginal on the precision, $1/\sigma_\epsilon$.

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

FTO Posterior marginals



FTO Posterior marginals



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))
names(lin.mod)  # Type this to see what can be extracted
##  [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"
```

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

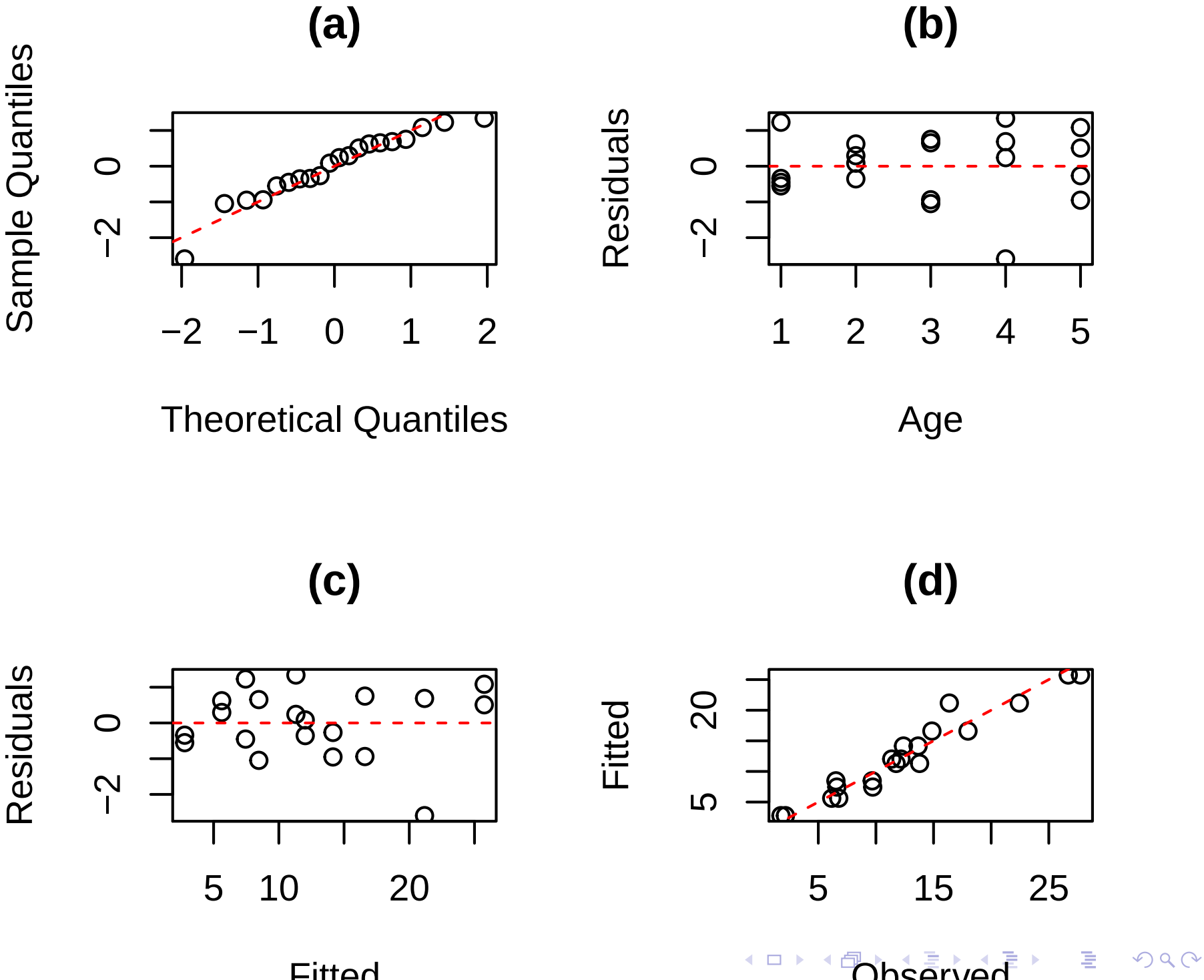
The code below forms residuals and then forms

- 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

Bayes logistic regression

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
##          0.4408          0.0577          0.0603          0.5587
##
## Fixed effects:
##                mean     sd 0.025quant 0.5quant 0.975quant    mode kld
## (Intercept) -1.8069 0.4554    -2.7495  -1.7900    -0.9593 -1.7555   0
## x            0.4799 0.2505     0.0083   0.4725     0.9935  0.4573   0
##
## The model has no random effects
```

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
##
## $sigma
## [1] 0.9784688
plot(seq(0, 7, 0.1), dlnorm(seq(0, 7, 0.1), meanlog = lnprior$mu,
    sdlog = lnprior$sigma), type = "l", xlab = "x",
    ylab = "LogNormal Density")
```

Bayes logistic regression

```
# 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
##          0.5431          0.0831          0.0484          0.6746
##
## Fixed effects:
##                mean     sd 0.025quant 0.5quant 0.975quant    mode kld
## (Intercept) -1.3227 0.2896    -1.9011  -1.3194    -0.7632 -1.3127   0
## x            0.1986 0.1536    -0.1001   0.1977     0.5025  0.1957   0
##
## The model has no random effects
##
## The model has no hyperparameters
##
## Expected number of effective parameters(std dev): 1.441(0.00)
## Number of equivalent replicates : 2.082
##
```

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!

```
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] 0.2199104 0.1806412
```

`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
##          0.7334          0.0747          0.0559          0.8640
##
## Fixed effects:
##               mean     sd 0.025quant 0.5quant 0.975quant   mode kld
## (Intercept) 0.5976 0.0698     0.4577   0.5976     0.7375 0.5976   0
##
## Random effects:
## Name   Model
##  ind   IID model
##
## Model hyperparameters:
##                                          mean     sd 0.025quant 0.5quant
## Precision for the Gaussian observations 21.08  4.691     13.137    20.68
## Precision for ind                       36.64 23.864      9.737    30.65
##                                         0.975quant  mode
## Precision for the Gaussian observations      31.47 19.95
## Precision for ind                            98.91 21.64
##
## Expected number of effective parameters(std dev): 7.80(1.051)
```

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)
```

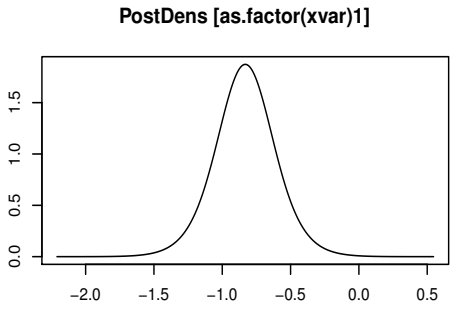
RNA Seq Analysis

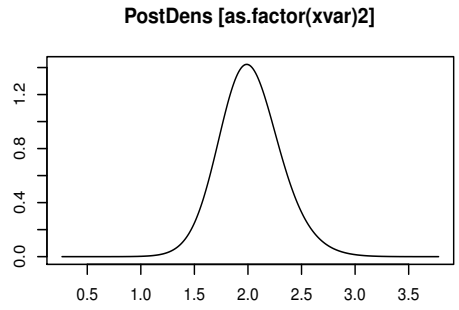
```
RNAfit$summary.fixed[1:10, c(1:5)]
##                             mean         sd  0.025quant    0.5quant 0.975quant
## as.factor(xvar)1  -0.83156203 0.2297015 -1.29229643 -0.83154256 -0.3710977
## as.factor(xvar)2   2.02186975 0.2929362  1.47472672  2.01016918  2.6360794
## as.factor(xvar)3   0.17295772 0.2558064 -0.33542775  0.17276183  0.6818231
## as.factor(xvar)4   0.43017762 0.2315997 -0.03422863  0.43027922  0.8937915
## as.factor(xvar)5   0.59644145 0.2302967  0.13473238  0.59642167  1.0580879
## as.factor(xvar)6   0.54559552 0.2346333  0.07590452  0.54560545  1.0149475
## as.factor(xvar)7   1.36315252 0.2825112  0.81263868  1.35994306  1.9319952
## as.factor(xvar)8   0.02797289 0.2306884 -0.43448440  0.02796718  0.4902795
## as.factor(xvar)9   0.01493844 0.2340923 -0.45375199  0.01494559  0.4833270
## as.factor(xvar)10  0.14968497 0.2405072 -0.33037102  0.14954191  0.6301853
RNAfit$summary.hyperpar[c(1:5)]
##                            mean      sd 0.025quant 0.5quant 0.975quant
## Precision for rep1 11.66733 6.85206   3.209893 10.14207   29.16818
```

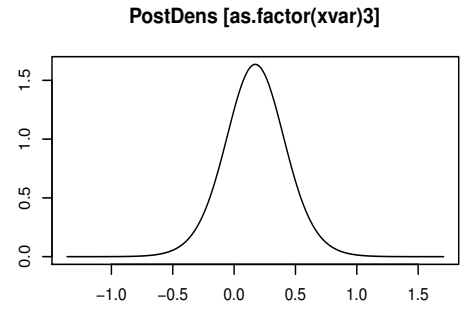
RNA Seq Analysis

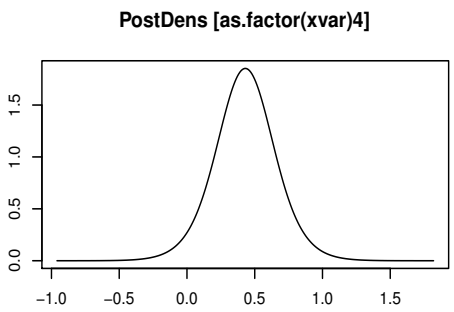
```
RNAfit$summary.random$rep1[, c(2:6)]
##            mean        sd  0.025quant    0.5quant 0.975quant
## 1  -0.22465618 0.2296986 -0.68569595 -0.22452256 0.23545613
## 2   0.22456889 0.2296985 -0.23572398  0.22442646 0.68551490
## 3  -0.46802312 0.2859345 -1.08495487 -0.45043090 0.05026434
## 4   0.46823802 0.2859737 -0.04943695  0.45064383 1.08604937
## 5  -0.18410619 0.2507125 -0.69351083 -0.18068348 0.30577506
## 6   0.18412438 0.2507133 -0.30591274  0.18069258 0.69416455
## 7   0.32963260 0.2315647 -0.13223191  0.32882968 0.79592176
## 8  -0.32958748 0.2315645 -0.79576846 -0.32880541 0.13187795
## 9  -0.14887881 0.2302828 -0.61106572 -0.14871323 0.31218531
## 10  0.14894141 0.2302829 -0.31228504  0.14875265 0.61107854
## 11  0.10007323 0.2343395 -0.36768653  0.09955925 0.57056509
## 12 -0.10001597 0.2343393 -0.57053854 -0.09952423 0.36738567
## 13 -0.18903550 0.2671270 -0.73888970 -0.18280446 0.32509506
## 14  0.18917908 0.2671381 -0.32408710  0.18294796 0.73940988
## 15  0.11139527 0.2306630 -0.35054008  0.11121265 0.57419783
## 16 -0.11139233 0.2306630 -0.57427177 -0.11122718 0.35035565
## 17 -0.05154561 0.2338444 -0.52043653 -0.05131788 0.41577325
## 18  0.05154718 0.2338443 -0.41604699  0.05130216 0.52031581
## 19 -0.06576912 0.2393472 -0.54600020 -0.06514843 0.41045745
## 20  0.06578486 0.2393477 -0.41094741  0.06514589 0.54593172
```

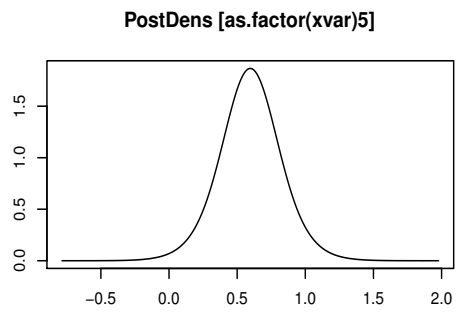
RNA Seq Analysis

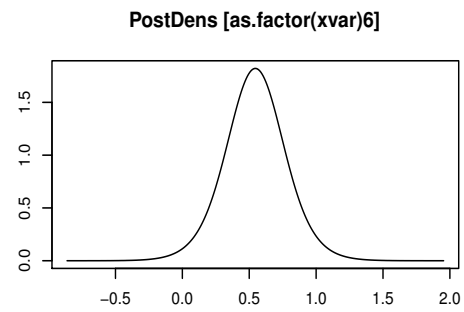


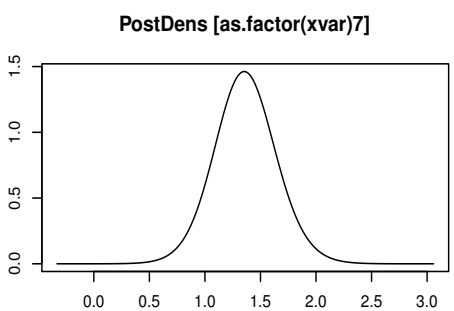


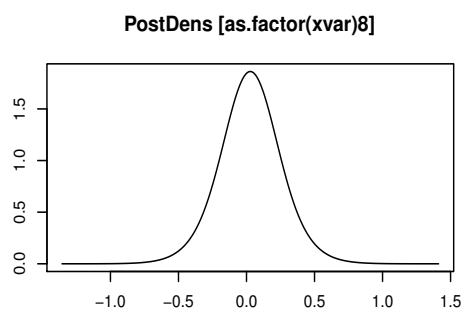


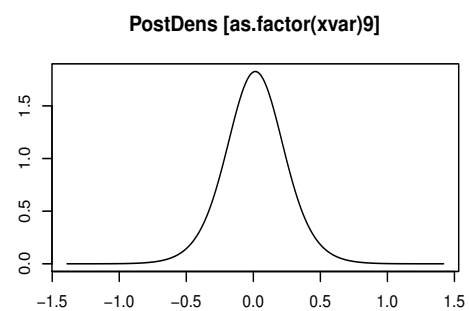




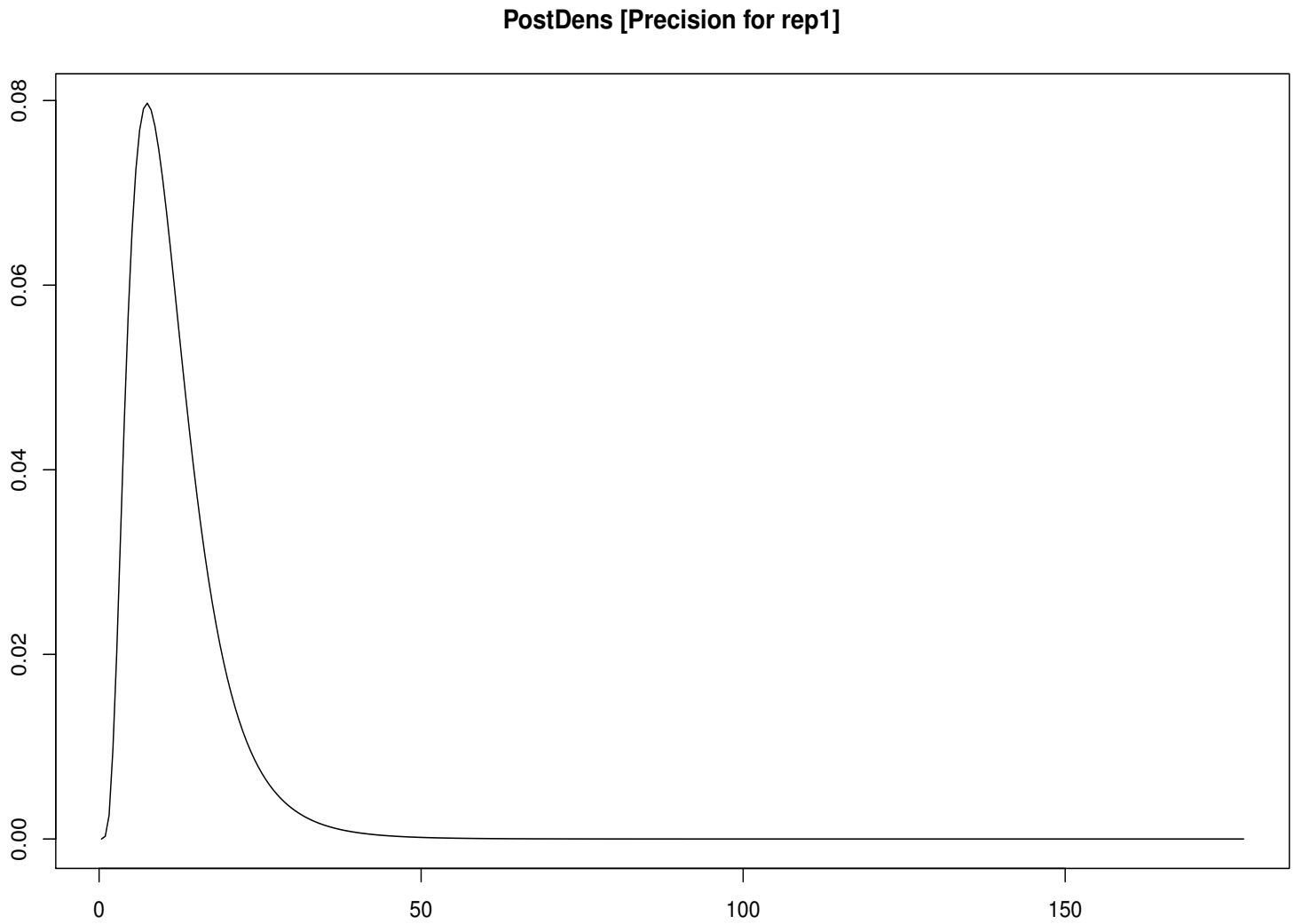




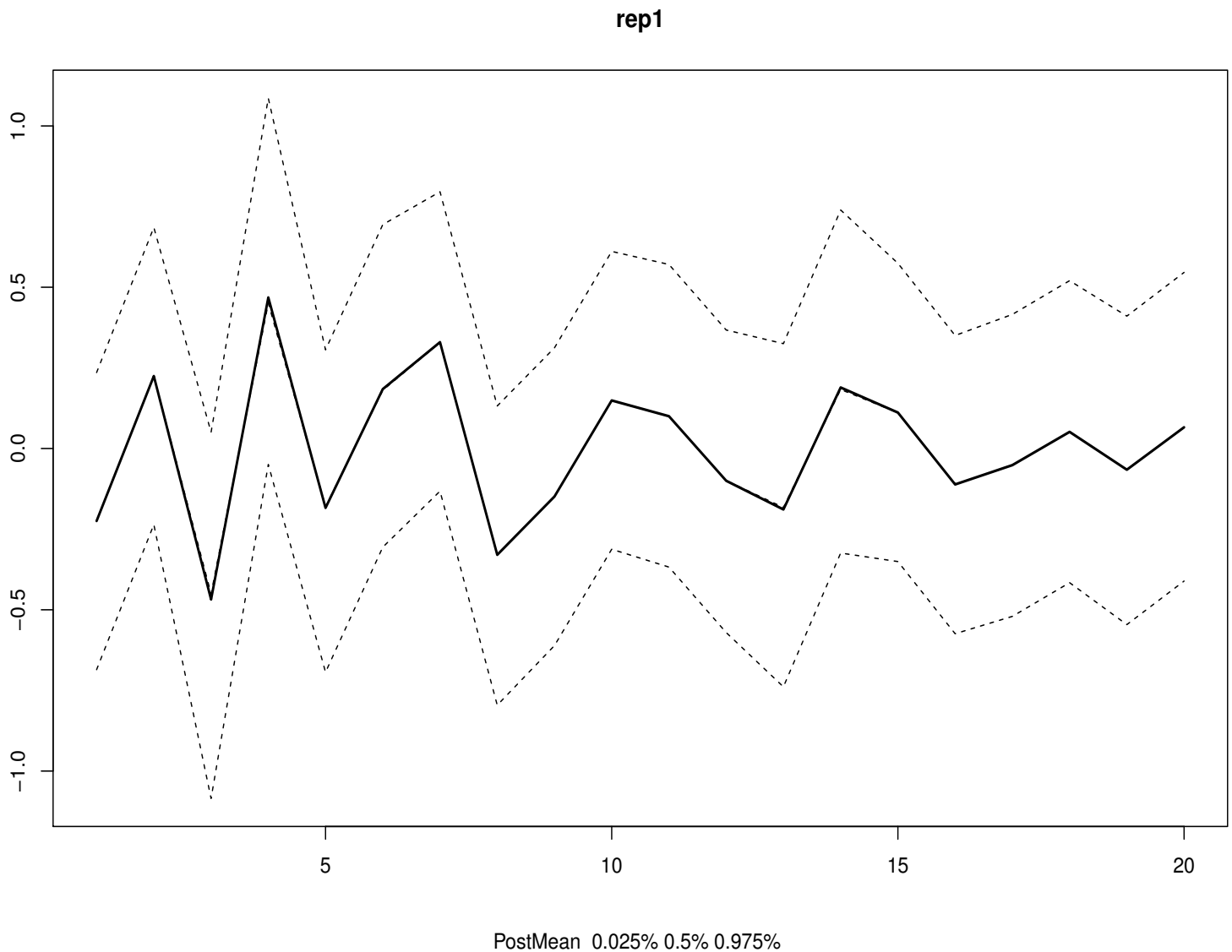




RNA Seq Analysis



RNA Seq Analysis



RNA Seq Analysis

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

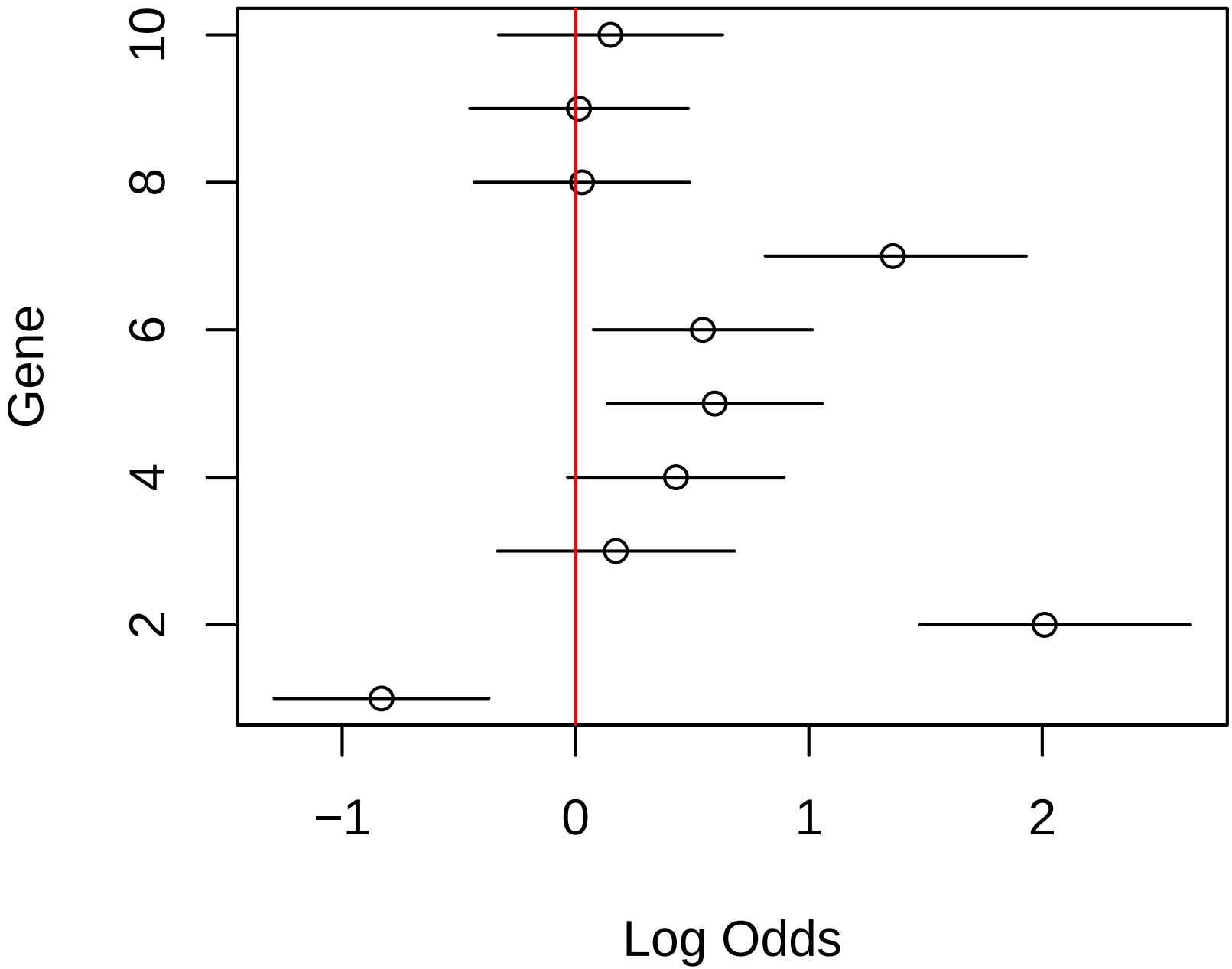
Comparison with 0 gives an indication of cis effects.

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



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).