Bayesian Computation with R Scripts

Jim Albert

2020-12-18

Contents

Chapter 1

Preface

This book contains all of the R scripts and associated output for Chapters 2 through 10 of *Bayesian Computation with R* second edition.

In these scripts, I have avoided the use of the `attach()` function and spaces have been added to increase readability.

Chapter 2

Introduction to Bayesian Thinking

2.1 Learning About the Proportion of Heavy Sleepers

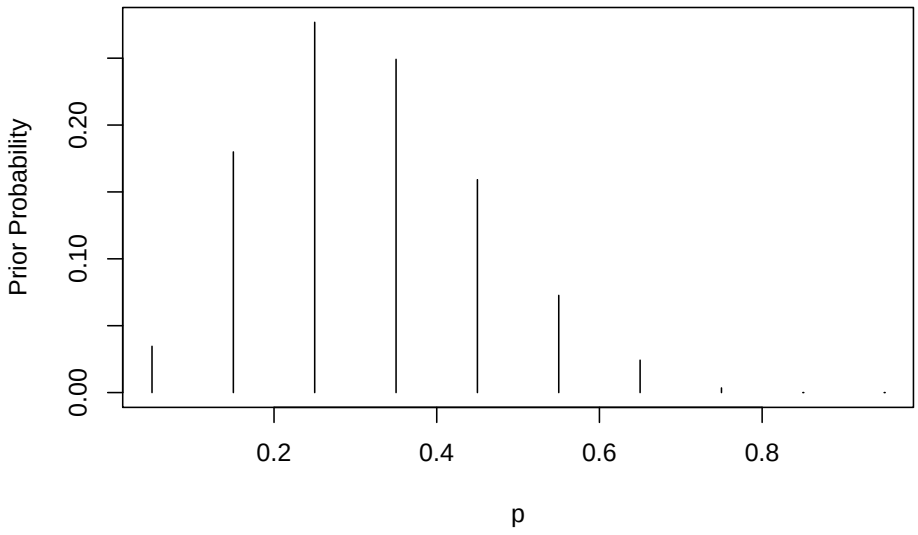
Want to learn about p, the proportion of heavy sleepers. Take a sample of 27 students and 11 are heavy sleepers.

2.2 Using a Discrete Prior

```
library(LearnBayes)
```

The prior for p:

```
p <- seq(0.05, 0.95, by = 0.1)
prior <- c(1, 5.2, 8, 7.2, 4.6, 2.1,
           0.7, 0.1, 0, 0)
prior <- prior / sum(prior)
plot(p, prior, type = "h",
     ylab="Prior Probability")
```



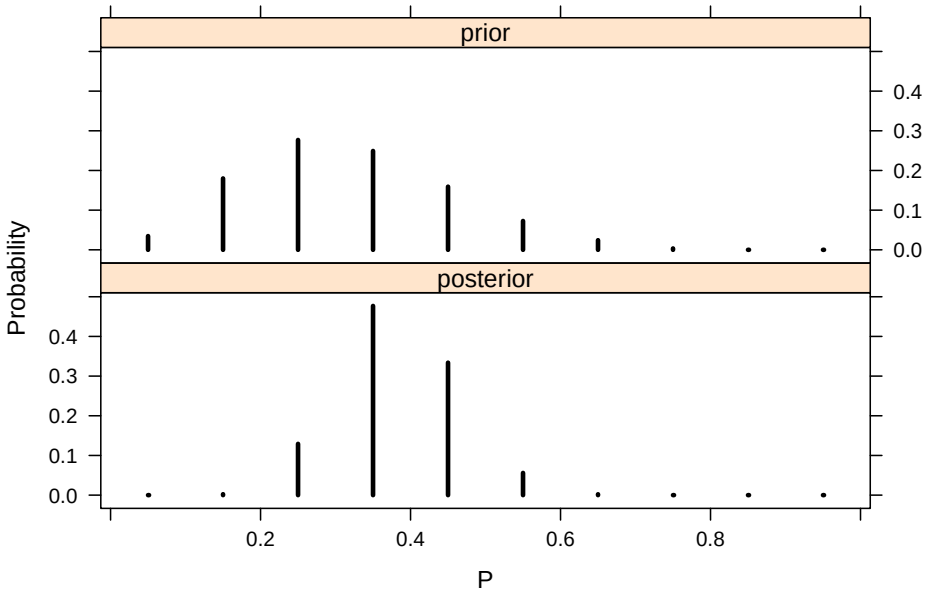
The posterior for p:

```
data <- c(11, 16)
post <- pdisc(p, prior, data)
round(cbind(p, prior, post),2)
```

```
##          p prior post
##  [1,] 0.05  0.03 0.00
##  [2,] 0.15  0.18 0.00
##  [3,] 0.25  0.28 0.13
##  [4,] 0.35  0.25 0.48
##  [5,] 0.45  0.16 0.33
##  [6,] 0.55  0.07 0.06
##  [7,] 0.65  0.02 0.00
##  [8,] 0.75  0.00 0.00
##  [9,] 0.85  0.00 0.00
## [10,] 0.95  0.00 0.00
```

```
library(lattice)
PRIOR <- data.frame("prior", p, prior)
POST <- data.frame("posterior", p, post)
names(PRIOR) <- c("Type", "P", "Probability")
names(POST) <- c("Type","P","Probability")
data <- rbind(PRIOR, POST)
```

```
xyplot(Probability ~ P | Type, data=data,
       layout=c(1,2), type="h", lwd=3, col="black")
```



2.3 Using a Beta Prior

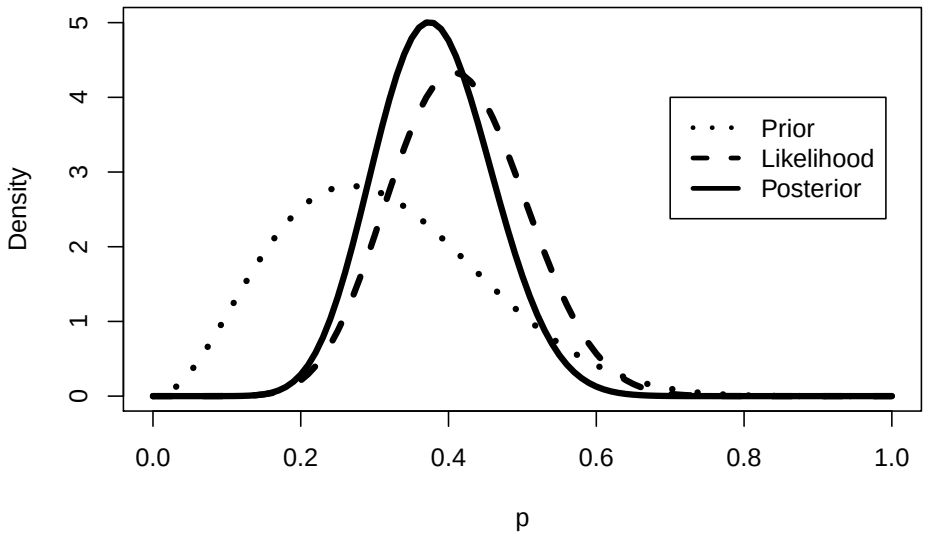
Construct a beta prior for p by inputting two percentiles:

```
quantile2 <- list(p=.9, x=.5)
quantile1 <- list(p=.5, x=.3)
(ab <- beta.select(quantile1,quantile2))
```

```
## [1] 3.26 7.19
```

Bayesian triplot:

```
a <- ab[1]
b <- ab[2]
s <- 11
f <- 16
curve(dbeta(x, a + s, b + f), from=0, to=1,
       xlab="p", ylab="Density", lty=1, lwd=4)
curve(dbeta(x, s + 1, f + 1), add=TRUE,
       lty=2, lwd=4)
curve(dbeta(x, a, b), add=TRUE, lty=3, lwd=4)
legend(.7, 4, c("Prior", "Likelihood",
                "Posterior"),
       lty=c(3, 2, 1), lwd=c(3, 3, 3))
```



Posterior summaries:

```
1 - pbeta(0.5, a + s, b + f)
```

```
## [1] 0.0690226
```

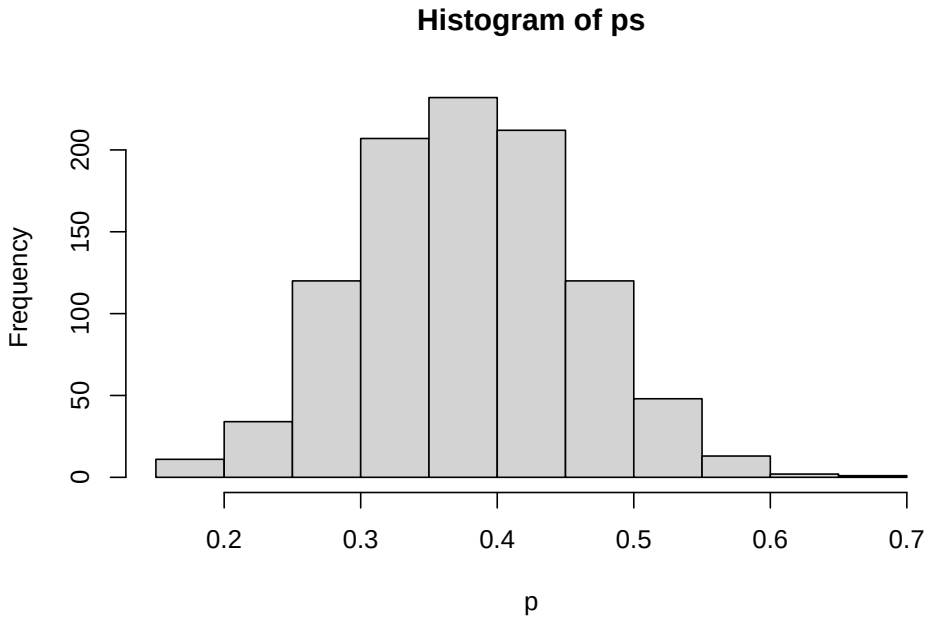
```
qbeta(c(0.05, 0.95), a + s, b + f)
```

```
## [1] 0.2555267 0.5133608
```

Simulating from posterior:

```
ps <- rbeta(1000, a + s, b + f)
```

```
hist(ps, xlab="p")
```




```
sum(ps >= 0.5) / 1000
```

```
## [1] 0.064
```

```
quantile(ps, c(0.05, 0.95))
```

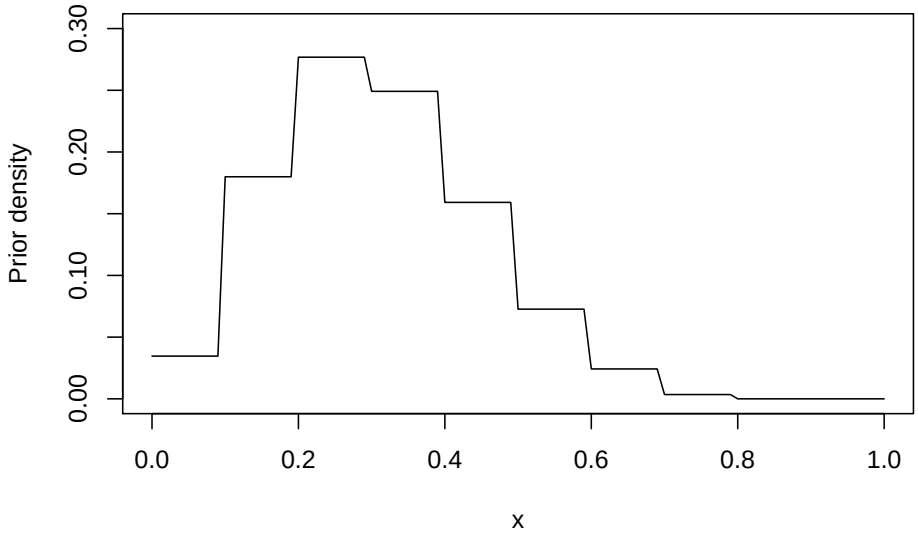
```
##        5%       95%
## 0.2530540 0.5124985
```

2.4 Using a Histogram Prior

Beliefs about p are expressed by a histogram prior. Illustrate brute force method of computing the posterior.

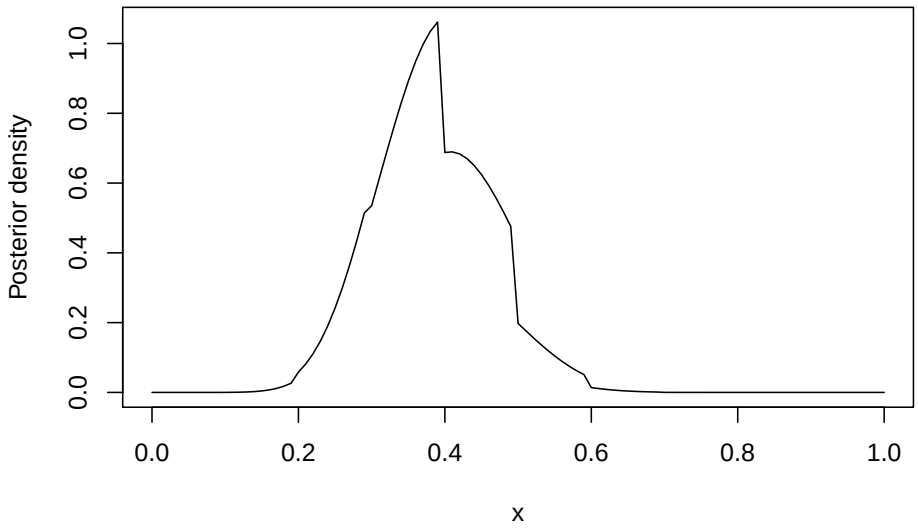
```
midpt <- seq(0.05, 0.95, by = 0.1)
prior <- c(1, 5.2, 8, 7.2, 4.6, 2.1, 0.7,
           0.1, 0, 0)
prior <- prior / sum(prior)
```

```
curve(histprior(x, midpt, prior), from=0, to=1,
   ylab="Prior density", ylim=c(0, .3))
```



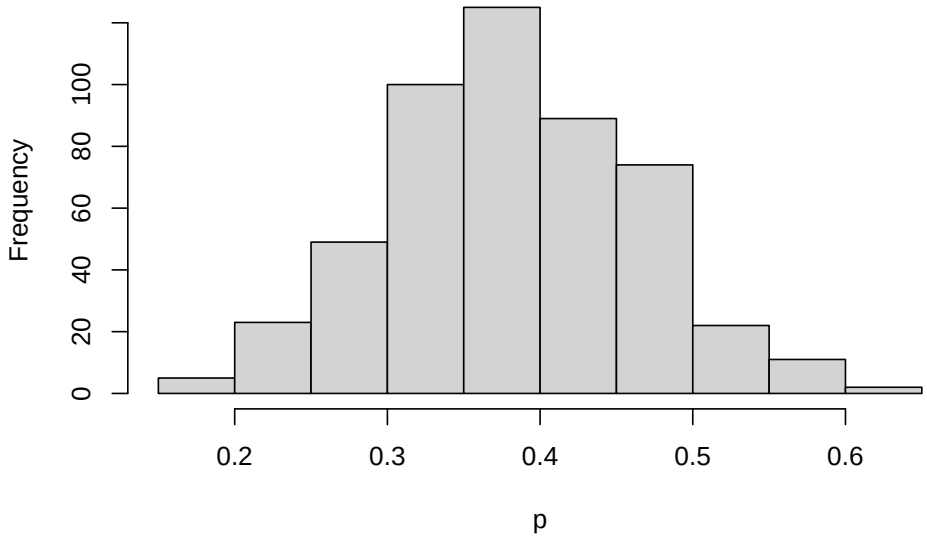

```
s <- 11
f <- 16
```

```
curve(histprior(x,midpt,prior) *
        dbeta(x, s + 1, f + 1),
   from=0, to=1, ylab="Posterior density")
```




```
p <- seq(0, 1, length=500)
post <- histprior(p, midpt, prior) *
        dbeta(p, s + 1, f + 1)
post <- post / sum(post)
ps <- sample(p, replace = TRUE, prob = post)
```

```
hist(ps, xlab="p", main="")
```



2.5 Prediction

Want to predict the number of heavy sleepers in a future sample of 20.

Discrete prior approach:

```
p <- seq(0.05, 0.95, by=.1)
prior <- c(1, 5.2, 8, 7.2, 4.6,
           2.1, 0.7, 0.1, 0, 0)
prior <- prior / sum(prior)
m <- 20
ys <- 0:20
pred <- pdiscp(p, prior, m, ys)
cbind(0:20, pred)
```

```
##                  pred
##  [1,]  0 2.030242e-02
##  [2,]  1 4.402694e-02
##  [3,]  2 6.894572e-02
##  [4,]  3 9.151046e-02
##  [5,]  4 1.064393e-01
##  [6,]  5 1.124487e-01
##  [7,]  6 1.104993e-01
##  [8,]  7 1.021397e-01
##  [9,]  8 8.932837e-02
## [10,]  9 7.416372e-02
```

```
## [11,] 10 5.851740e-02
## [12,] 11 4.383668e-02
## [13,] 12 3.107700e-02
## [14,] 13 2.071698e-02
## [15,] 14 1.284467e-02
## [16,] 15 7.277453e-03
## [17,] 16 3.667160e-03
## [18,] 17 1.575535e-03
## [19,] 18 5.381536e-04
## [20,] 19 1.285179e-04
## [21,] 20 1.584793e-05
```

Continuous prior approach:

```
ab <- c(3.26, 7.19)
m <- 20
ys <- 0:20
pred <- pbetap(ab, m, ys)
```

Simulating predictive distribution:

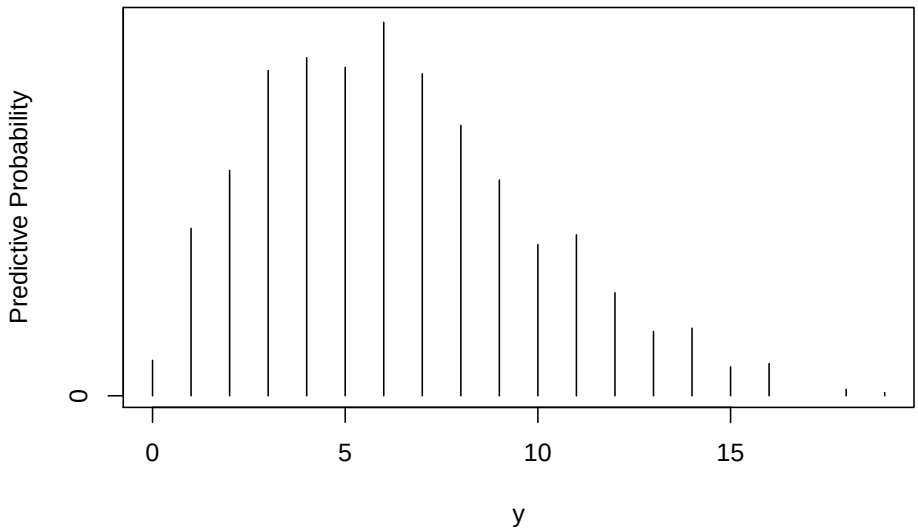
```
p <- rbeta(1000, 3.26, 7.19)
```

```
y <- rbinom(1000, 20, p)
```

```
table(y)
```

```
## y
##   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  18  19
##  11  52  70 101 105 102 116 100  84  67  47  50  32  20  21   9  10   2   1
```

```
freq <- table(y)
ys <- as.integer(names(freq))
predprob <- freq / sum(freq)
plot(ys, predprob, type="h", xlab="y",
   ylab="Predictive Probability")
```




```
dist <- cbind(ys, predprob)
```

Construction of a prediction interval:

```
covprob <- .9
discint(dist, covprob)
```

```
## $prob
##    12
## 0.926
##
## $set
##  1  2  3  4  5  6  7  8  9 10 11 12
##  1  2  3  4  5  6  7  8  9 10 11 12
```

Chapter 3

Single-Parameter Models

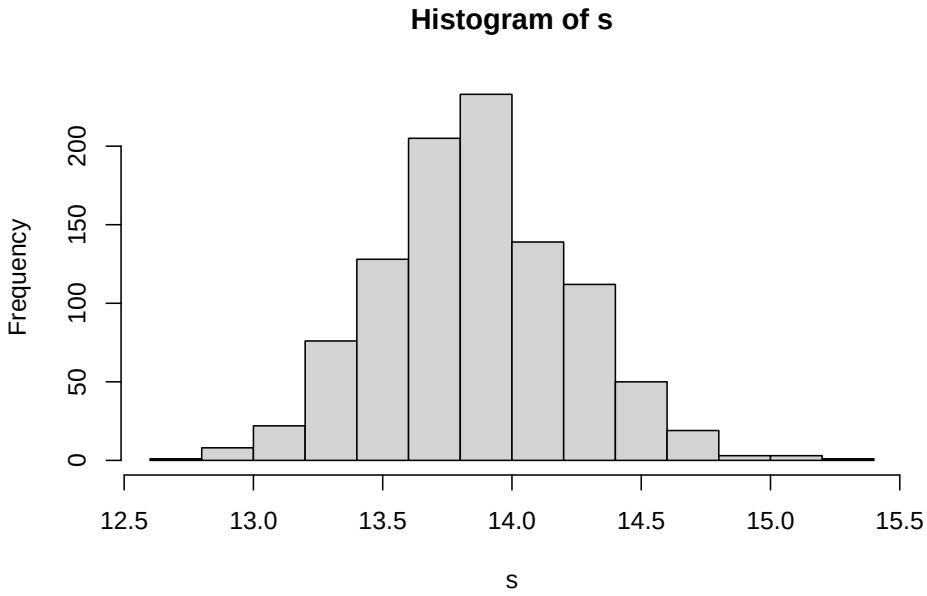
3.1 Normal Distribution with Known Mean but Unknown Variance

Assuming we have a sample $\{y_j\}$ from a normal distribution with mean 0 and variance σ^2. Assuming the prior $g(\sigma^2) \propto 1/\sigma^2$, simulating from the posterior.

```
library(LearnBayes)
```

```
d <- with(footballscores,
          favorite - underdog - spread)
n <- length(d)
v <- sum(d ^ 2)
```

```
P <- rchisq(1000, n) / v
s <- sqrt(1 / P)
hist(s)
```




```
quantile(s, probs = c(0.025, 0.5, 0.975))
```

```
##     2.5%      50%    97.5%
## 13.16744 13.85293 14.60743
```

3.2 Estimating a Heart Transplant Mortality Rate

Have a sample $\{y_j\}$ from a Poisson($e\lambda$) distribution where the exposure e is known. Assigning λ a gamma(α, β) prior.

Predictive density:

```
alpha <- 16; beta <- 15174
yobs <- 1; ex <- 66
y <- 0:10
lam <- alpha / beta
py <- dpois(y, lam * ex) *
  dgamma(lam, shape = alpha, rate = beta) /
  dgamma(lam, shape = alpha + y, rate = beta + ex)
cbind(y, round(py, 3))
```

```
##        y
##  [1,]  0 0.933
##  [2,]  1 0.065
##  [3,]  2 0.002
##  [4,]  3 0.000
```

```
##  [5,]  4 0.000
##  [6,]  5 0.000
##  [7,]  6 0.000
##  [8,]  7 0.000
##  [9,]  8 0.000
## [10,]  9 0.000
## [11,] 10 0.000
```

Posterior density:

```
lambdaA <- rgamma(1000, shape = alpha + yobs,
                  rate = beta + ex)
```

Data from a different hospital:

```
ex <- 1767; yobs <-4
y <- 0:10
py <- dpois(y, lam * ex) *
  dgamma(lam, shape = alpha, rate = beta) /
  dgamma(lam, shape = alpha + y, rate = beta + ex)
 cbind(y, round(py, 3))
```

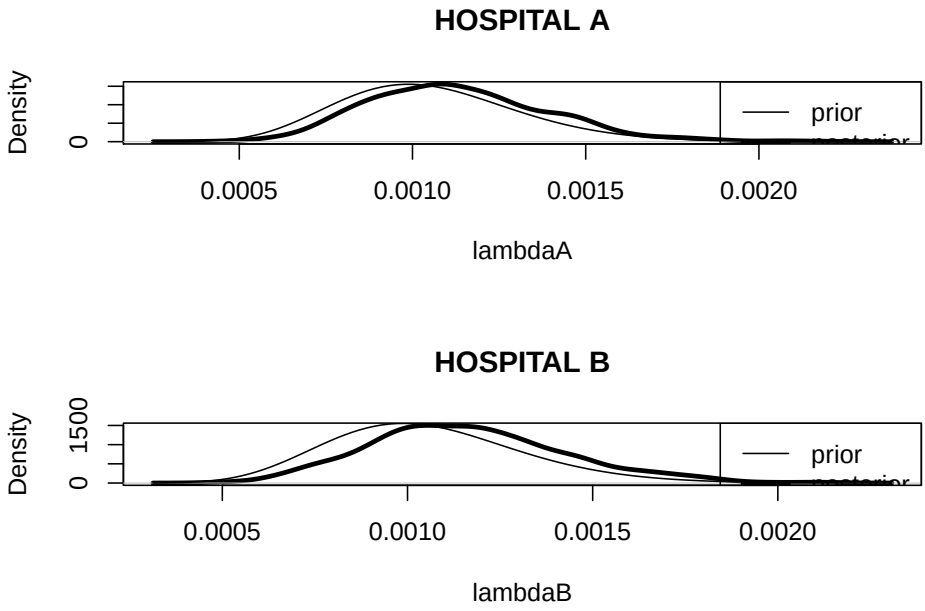
```
##        y
##  [1,]  0 0.172
##  [2,]  1 0.286
##  [3,]  2 0.254
##  [4,]  3 0.159
##  [5,]  4 0.079
##  [6,]  5 0.033
##  [7,]  6 0.012
##  [8,]  7 0.004
##  [9,]  8 0.001
## [10,]  9 0.000
## [11,] 10 0.000
```

```
lambdaB <- rgamma(1000, shape = alpha + yobs,
                  rate = beta + ex)
```

Prior and posteriors for two hospitals:

```
par(mfrow = c(2, 1))
 plot(density(lambdaA), main="HOSPITAL A",
      xlab="lambdaA", lwd=3)
 curve(dgamma(x, shape = alpha, rate = beta),
       add=TRUE)
 legend("topright",legend=c("prior","posterior"),
        lwd=c(1,3))
 plot(density(lambdaB), main="HOSPITAL B",
      xlab="lambdaB", lwd=3)
```

```
curve(dgamma(x, shape = alpha, rate = beta),
      add=TRUE)
legend("topright",legend=c("prior","posterior"),
       lwd=c(1,3))
```



3.3 An Illustration of Bayesian Robustness

Assuming normal sampling (known standard deviation), compare the use of two priors on the mean μ.

```
quantile1 <- list(p=.5, x=100)
quantile2 <- list(p=.95, x=120)
normal.select(quantile1, quantile2)
```

```
## $mu
## [1] 100
##
## $sigma
## [1] 12.15914
```

```
mu <- 100
tau <- 12.16
sigma <- 15
n <- 4
se <- sigma / sqrt(4)
ybar <- c(110, 125, 140)
```

```
tau1 <- 1 / sqrt(1 / se ^ 2 + 1 / tau ^ 2)
mu1 <- (ybar / se ^ 2 + mu / tau ^ 2) * tau1 ^ 2
summ1 <- cbind(ybar, mu1, tau1)
summ1
```

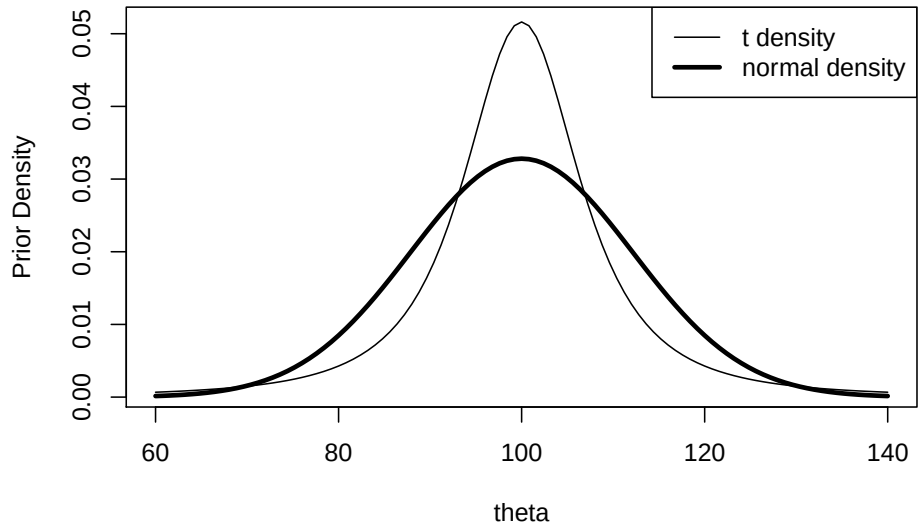
```
##      ybar      mu1     tau1
## [1,]  110 107.2442 6.383469
## [2,]  125 118.1105 6.383469
## [3,]  140 128.9768 6.383469
```

Compare two possible priors for μ:

```
tscale <- 20 / qt(0.95, 2)
tscale
```

```
## [1] 6.849349
```

```
par(mfrow=c(1, 1))
curve(1 / tscale * dt((x - mu) / tscale, 2),
   from=60, to=140, xlab="theta",
   ylab="Prior Density")
curve(dnorm(x, mean=mu, sd=tau), add=TRUE, lwd=3)
legend("topright", legend=c("t density",
                            "normal density"),
       lwd=c(1,3))
```




```
norm.t.compute <- function(ybar){
     theta <- seq(60, 180, length = 500)
     like <- dnorm(theta, mean=ybar,
                   sd=sigma/sqrt(n))
     prior <- dt((theta - mu) / tscale, 2)
```

```
    post <- prior * like
    post <- post / sum(post)
    m <- sum(theta * post)
    s <- sqrt(sum(theta ^ 2 * post) - m ^ 2)
    c(ybar, m, s)
}
```

```
summ2 <- t(sapply(c(110, 125, 140),
                  norm.t.compute))
dimnames(summ2)[[2]] <- c("ybar", "mu1 t",
                          "tau1 t")
summ2
```

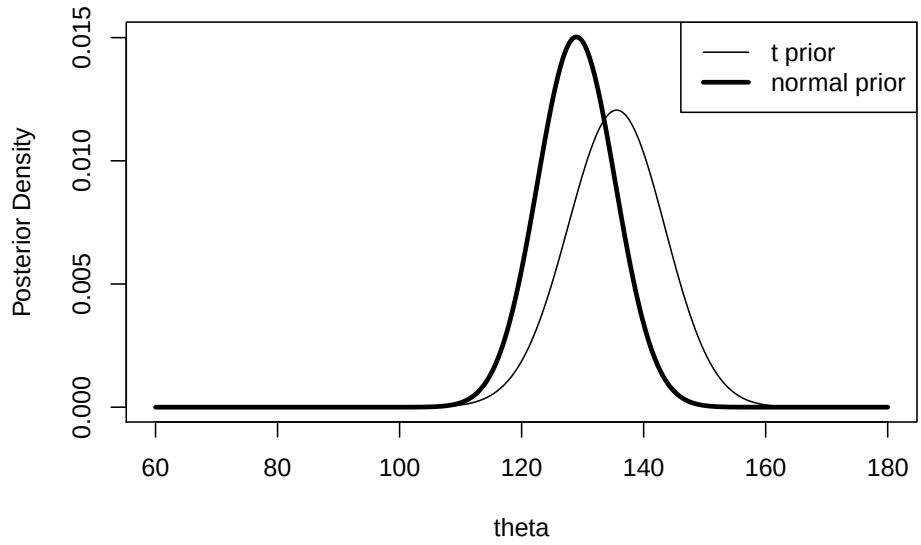
```
##      ybar    mu1 t   tau1 t
## [1,]  110 105.2921 5.841676
## [2,]  125 118.0841 7.885174
## [3,]  140 135.4134 7.973498
```

```
cbind(summ1, summ2)
```

```
##      ybar      mu1     tau1 ybar    mu1 t   tau1 t
## [1,]  110 107.2442 6.383469  110 105.2921 5.841676
## [2,]  125 118.1105 6.383469  125 118.0841 7.885174
## [3,]  140 128.9768 6.383469  140 135.4134 7.973498
```

Compare two posterior densities:

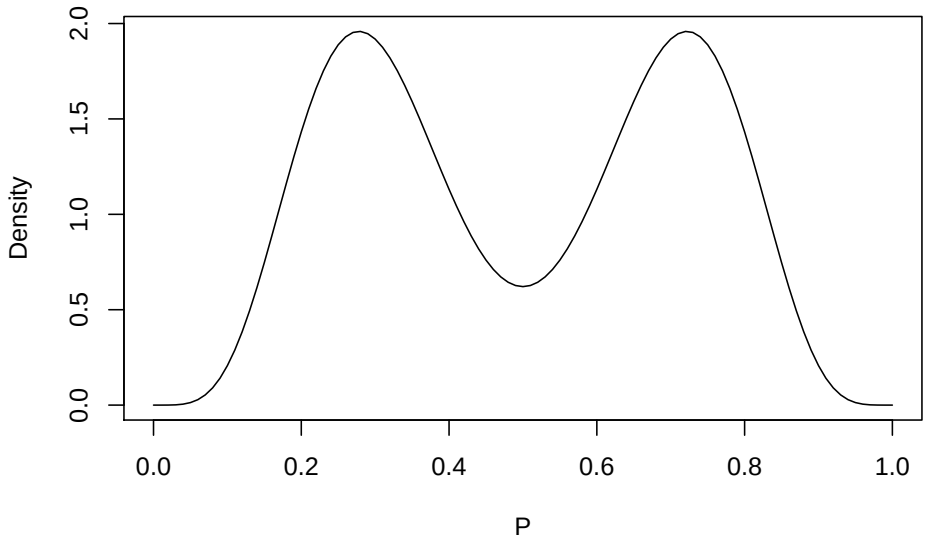
```
theta <- seq(60, 180, length=500)
normpost <- dnorm(theta, mu1[3], tau1)
normpost <- normpost / sum(normpost)
plot(theta, normpost, type="l", lwd=3,
     ylab="Posterior Density")
like <- dnorm(theta, mean=140, sd=sigma / sqrt(n))
prior <- dt((theta - mu) / tscale, 2)
tpost <- prior * like / sum(prior * like)
lines(theta, tpost)
legend("topright", legend=c("t prior",
                            "normal prior"),
       lwd=c(1,3))
```



3.4 Mixtures of Conjugate Priors

Use a mixture of beta curves to reflect beliefs that a particular coin is biased.

```
curve(.5 * dbeta(x, 6, 14) + .5 * dbeta(x, 14, 6),
      from=0, to=1, xlab="P", ylab="Density")
```



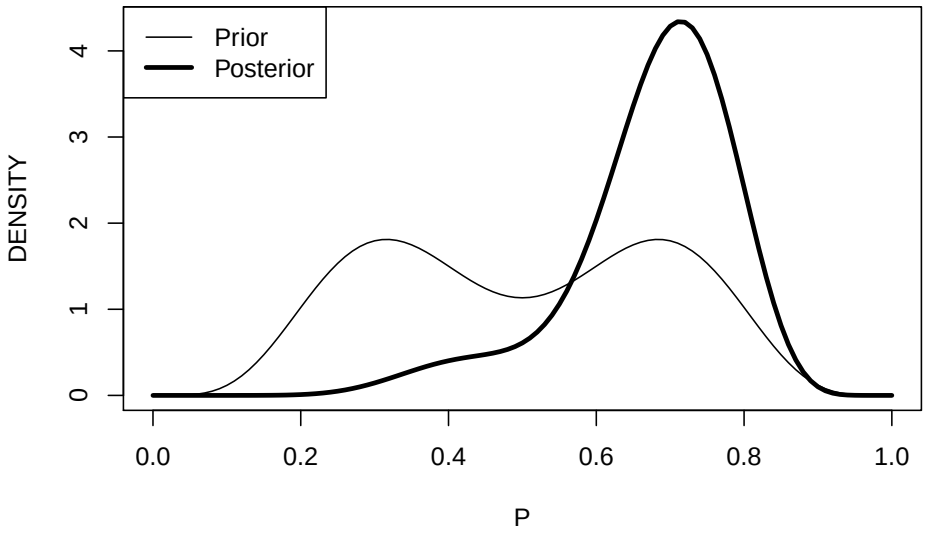

```
probs <- c(.5, .5)
beta.par1 <- c(6, 14)
beta.par2 <- c(14, 6)
betapar <- rbind(beta.par1, beta.par2)
```

```
data <- c(7, 3)
post <- binomial.beta.mix(probs, betapar, data)
post
```

```
## $probs
##  beta.par1  beta.par2
## 0.09269663 0.90730337
##
## $betapar
##           [,1] [,2]
## beta.par1   13   17
## beta.par2   21    9
```

Compare prior and posterior densities for the probability coin lands heads.

```
curve(post$probs[1] * dbeta(x,13,17) +
        post$probs[2] * dbeta(x,21,9),
  from=0, to=1, lwd=3,
  xlab="P", ylab="DENSITY")
curve(.5 * dbeta(x, 6, 12) +
        .5 * dbeta(x, 12, 6), 0, 1, add=TRUE)
legend("topleft", legend=c("Prior", "Posterior"),
       lwd=c(1, 3))
```



3.5 A Bayesian Test of the Fairness of a Coin

Testing if a coin is fair. Observe 5 heads in 20 flips.

P-value calculation:

```
pbinom(5, 20, 0.5)
```

```
## [1] 0.02069473
```

Bayesian test of fairness using a mixture prior.

```
n <- 20
y <- 5
a <- 10
p <- 0.5
m1 <- dbinom(y, n, p) * dbeta(p, a, a) /
  dbeta(p, a + y, a + n - y)
lambda <- dbinom(y, n, p) / (dbinom(y, n, p) + m1)
lambda
```

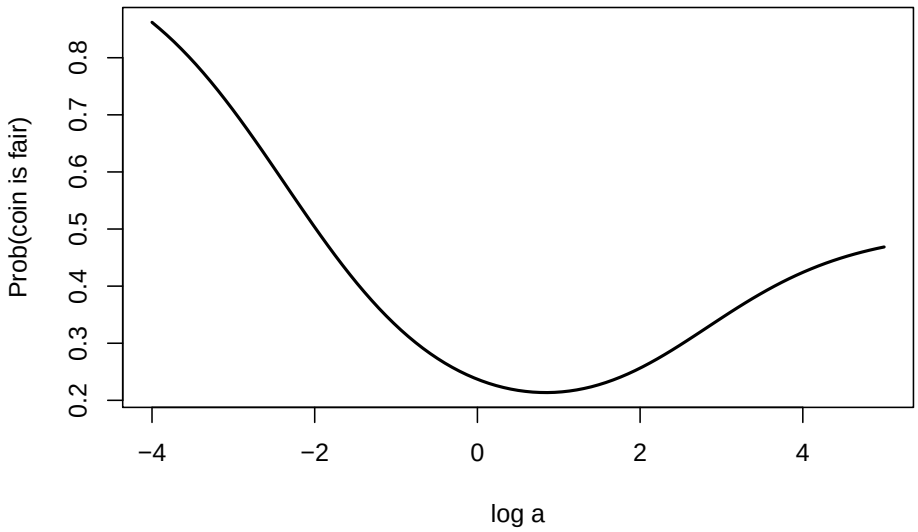
```
## [1] 0.2802215
```

```
pbetat(p, .5, c(a, a), c(y, n - y))
```

```
## $bf
## [1] 0.3893163
##
## $post
## [1] 0.2802215
```

```
prob.fair <- function(log.a){
  a <- exp(log.a)
  m2 <- dbinom(y, n, p) * dbeta(p, a, a) /
            dbeta(p, a + y, a + n - y)
  dbinom(y, n, p) / (dbinom(y, n, p) + m2)
}
```

```
n <- 20; y <- 5; p <- 0.5
curve(prob.fair(x), from = -4, to = 5,
      xlab="log a",
      ylab="Prob(coin is fair)", lwd=2)
```




```
n <- 20; y <- 5
a <- 10; p <- .5
m2 <- 0
 for (k in 0:y){
   m2 <- m2 + dbinom(k, n, p) * dbeta(p, a, a) /
     dbeta(p, a + k, a + n - k)
}
lambda <- pbinom(y, n, p) / (pbinom(y, n, p) + m2)
lambda
```

```
## [1] 0.2184649
```

Chapter 4

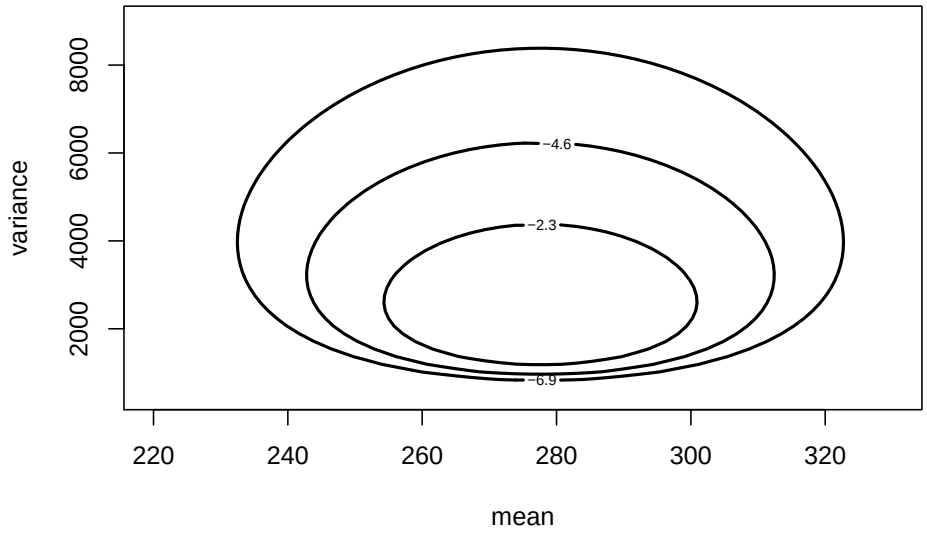
Multiparameter Models

4.1 Normal Data with Both Parameters Unknown

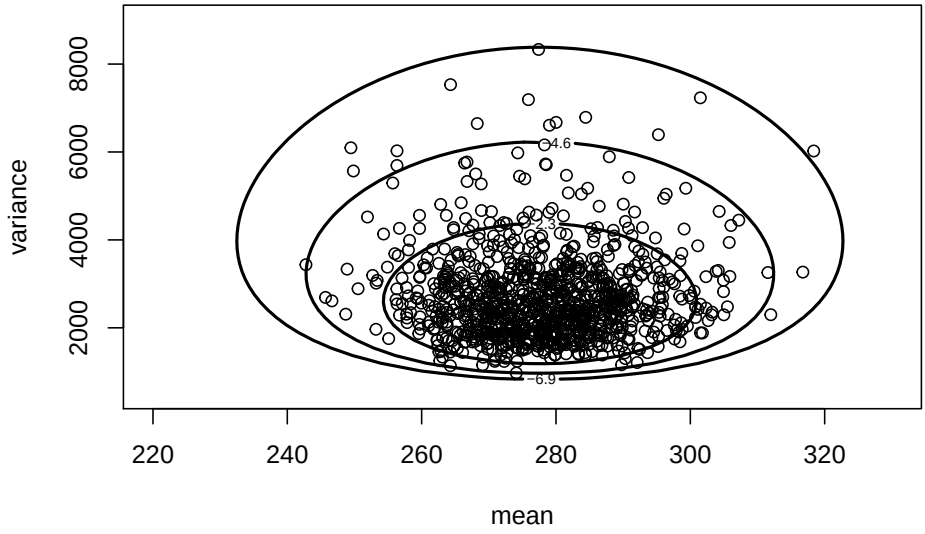
Illustrates exact posterior sampling of (μ, σ^2) for normal sampling with a non-informative prior.

```
library(LearnBayes)
```

```
mycontour(normchi2post,
                c(220, 330, 500, 9000),
                marathontimes$time,
                xlab="mean", ylab="variance")
```



```
S <- with(marathontimes,
          sum((time - mean(time))^2))
n <- length(marathontimes$time)
sigma2 <- S / rchisq(1000, n - 1)
mu <- rnorm(1000, mean = mean(marathontimes$time),
            sd = sqrt(sigma2) / sqrt(n))
mycontour(normchi2post,
               c(220, 330, 500, 9000),
               marathontimes$time,
               xlab="mean", ylab="variance")
points(mu, sigma2)
```




```
quantile(mu, c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 256.7045 301.1136
```

```
quantile(sqrt(sigma2), c(0.025, 0.975))
```

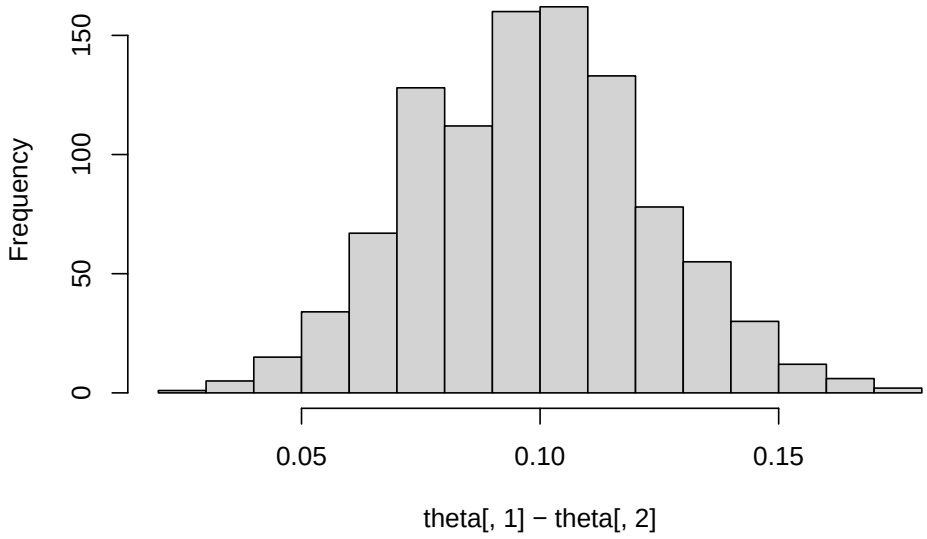
```
##     2.5%    97.5%
## 37.85306 73.41654
```

4.2 A Multinomial Model

Multinomial data and a uniform prior placed on the proportions. Sampling from the Dirichlet posterior distribution.

```
alpha <- c(728, 584, 138)
theta <- rdirichlet(1000, alpha)
```

```
hist(theta[, 1] - theta[, 2], main="")
```



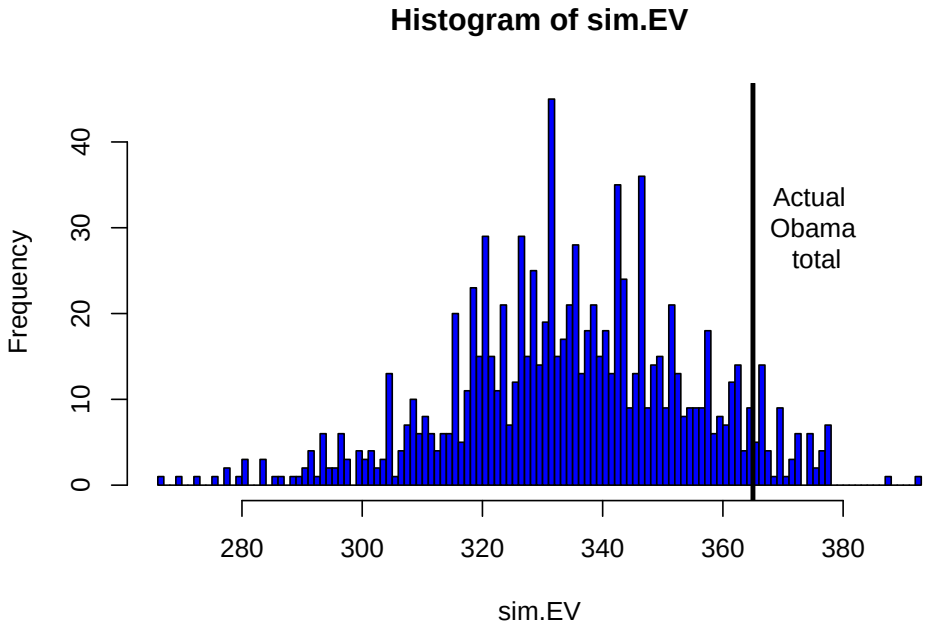
Considers posterior distribution of Obama electoral votes for the 2008 presidential election.

```
prob.Obama <- function(j){
  p <- with(election.2008,
            rdirichlet(5000,
          500 * c(M.pct[j], O.pct[j],
          100 - M.pct[j] - O.pct[j]) / 100 + 1))
  mean(p[, 2] > p[, 1])
}
Obama.win.probs <- sapply(1 : 51, prob.Obama)
```

```
sim.election <- function(){
  winner <- rbinom(51, 1,
              Obama.win.probs)
  sum(election.2008$EV * winner)
}
```

```
sim.EV <- replicate(1000, sim.election())
```

```
hist(sim.EV, min(sim.EV) : max(sim.EV), col="blue")
abline(v=365, lwd=3)  # Obama received 365 votes
text(375, 30, "Actual \n Obama \n total")
```



4.3 A Bioassay Experiment

Bayesian fitting of a logistic model using data from a dose-response experiment.

```
x <- c(-0.86, -0.3, -0.05, 0.73)
n <- c(5, 5, 5, 5)
y <- c(0, 1, 3, 5)
data <- cbind(x, n, y)
```

Traditional logistic model fit.

```
glmdata <- cbind(y, n - y)
results <- glm(glmdata ~ x, family = binomial)
summary(results)
```

```
##
## Call:
## glm(formula = glmdata ~ x, family = binomial)
##
## Deviance Residuals:
##        1         2         3         4
## -0.17236   0.08133  -0.05869   0.12237
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.8466     1.0191   0.831    0.406
```

```
## x             7.7488     4.8728   1.590    0.112
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 15.791412  on 3  degrees of freedom
## Residual deviance:  0.054742  on 2  degrees of freedom
## AIC: 7.9648
##
## Number of Fisher Scoring iterations: 7
```

Illustration of a conditional means prior. When x = -.7, median and 90th percentile of p are (.2,.4). When x = +.6, median and 90th percentile of p are (.8, .95)

```
a1.b1 <- beta.select(list(p=.5, x=.2),
                     list(p=.9, x=.5))
a2.b2 <- beta.select(list(p=.5, x=.8),
                    list(p=.9, x=.98))
```

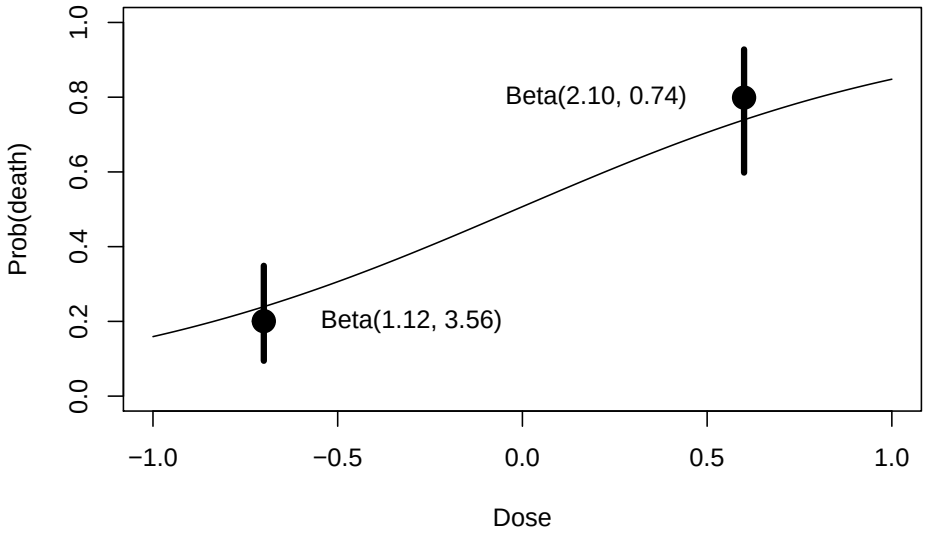
```
prior <- rbind(c(-0.7, 4.68, 1.12),
               c(0.6, 2.10, 0.74))
data.new <- rbind(data, prior)
```

Plot prior.

```
plot(c(-1,1), c(0, 1), type="n",
     xlab="Dose", ylab="Prob(death)")
lines(-0.7 * c(1, 1), qbeta(c(.25, .75),
      a1.b1[1], a1.b1[2]), lwd=4)
lines(0.6 * c(1, 1), qbeta(c(.25, .75),
      a2.b2[1], a2.b2[2]), lwd=4)
points(c(-0.7, 0.6), qbeta(.5, c(a1.b1[1],
          a2.b2[1]), c(a1.b1[2], a2.b2[2])),
         pch=19, cex=2)
text(-0.3, .2, "Beta(1.12, 3.56)")
text(.2, .8, "Beta(2.10, 0.74)")
response <- rbind(a1.b1, a2.b2)
x <- c(-0.7, 0.6)
fit <- glm(response ~ x, family = binomial)
```

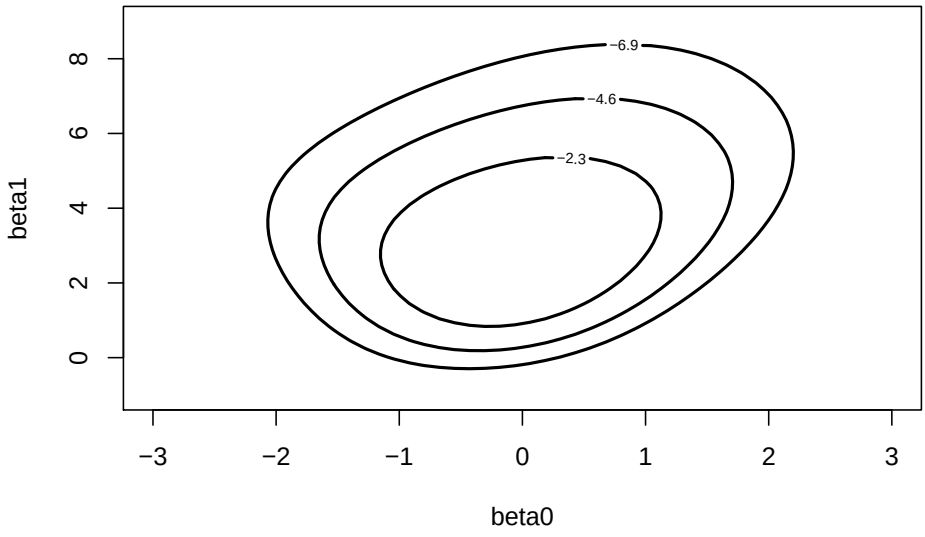
```
## Warning in eval(family$initialize): non-integer counts in a binomial glm!
```

```
curve(exp(fit$coef[1] + fit$coef[2] * x) /
     (1 + exp(fit$coef[1] + fit$coef[2] * x)),
     add=T)
```

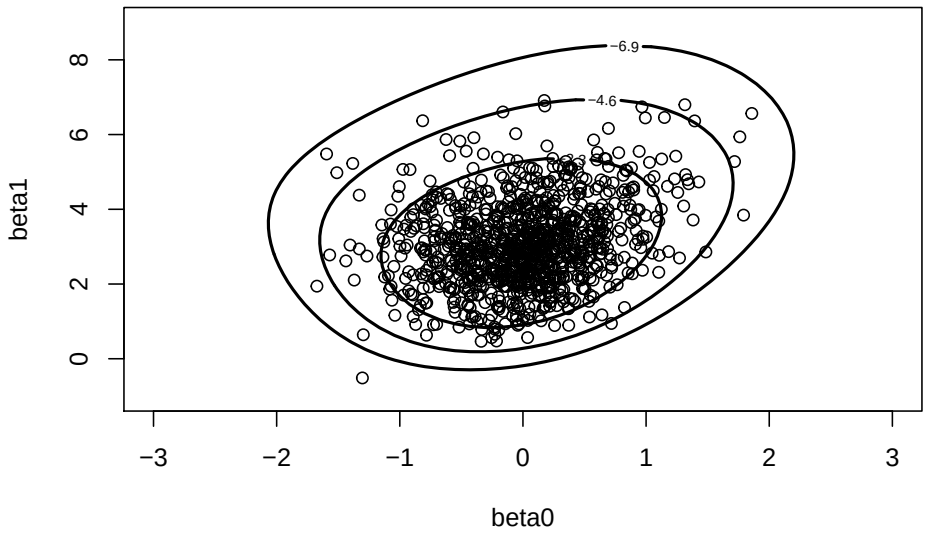


Posterior of regression coefficients.

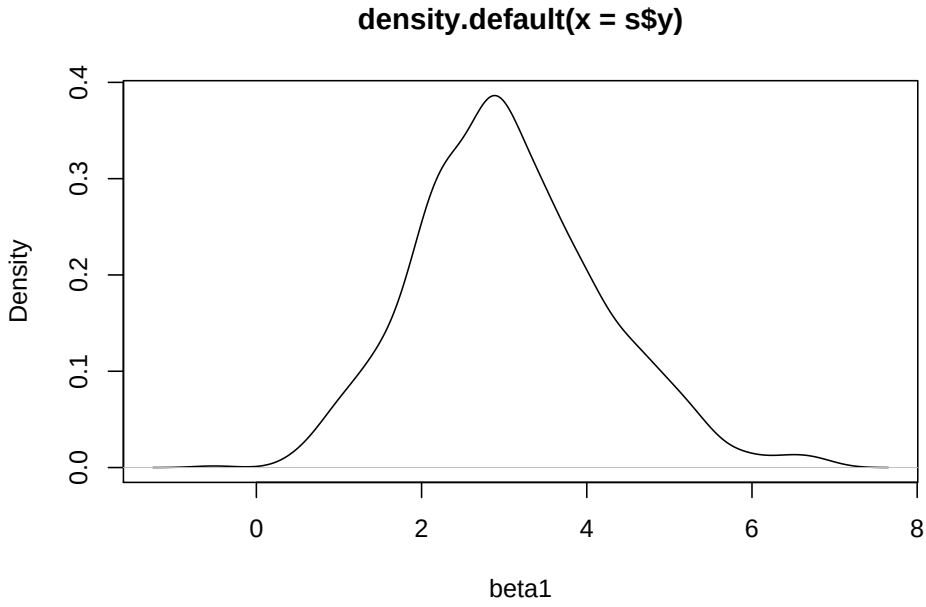
```
mycontour(logisticpost, c(-3, 3, -1, 9), data.new,
  xlab="beta0", ylab="beta1")
```




```
mycontour(logisticpost, c(-3, 3, -1, 9), data.new,
  xlab="beta0", ylab="beta1")
s <- simcontour(logisticpost, c(-2, 3, -1, 11),
                data.new, 1000)
points(s)
```

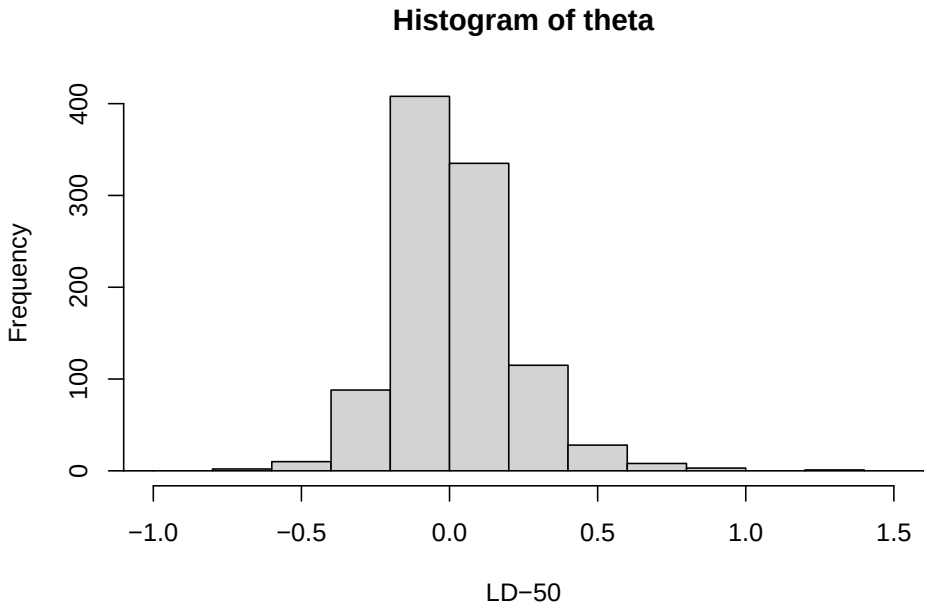



```
plot(density(s$y), xlab="beta1")
```



Estimation of LD50 parameter.

```
theta <- -s$x / s$y
hist(theta, xlab="LD-50", breaks=20,
     xlim = c(-1, 1.5))
```



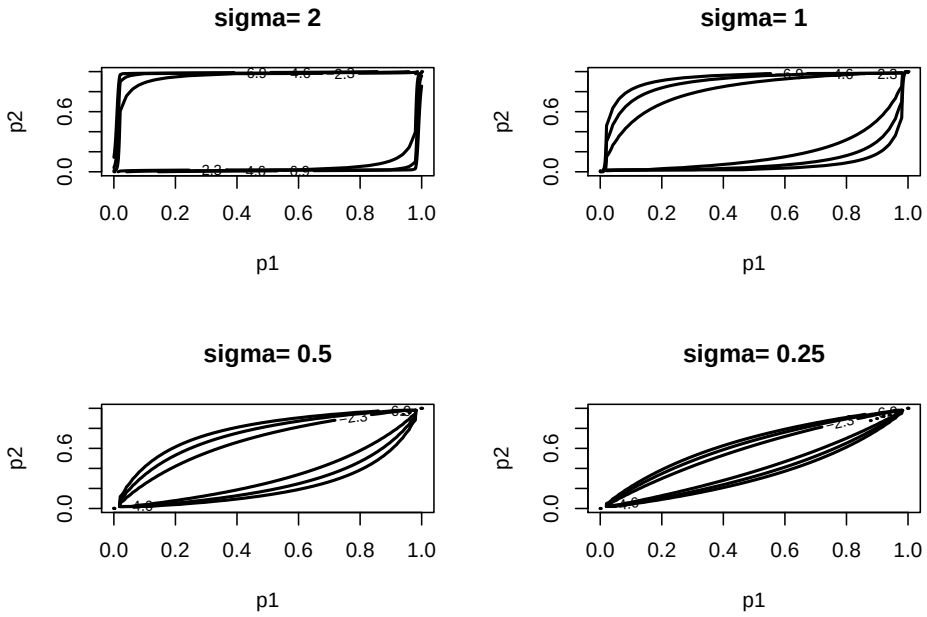

```
quantile(theta, c(.025, .975))
```

```
##       2.5%      97.5%
## -0.3194579  0.5101581
```

4.4 Comparing Two Proportions

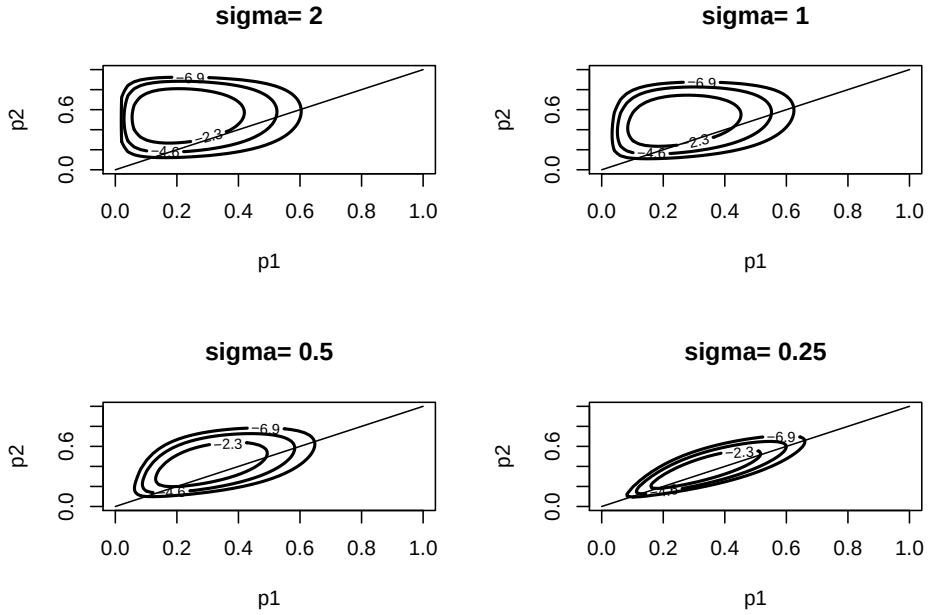
Using Howard's dependent prior for two proportions. Graph of the prior.

```
sigma <- c(2, 1, .5, .25)
plo <- .0001; phi <- .9999
par(mfrow=c(2, 2))
for (i in 1:4){
    mycontour(howardprior,
              c(plo, phi, plo, phi),
              c(1, 1, 1, 1, sigma[i]),
      main=paste("sigma=", as.character(sigma[i])),
      xlab="p1", ylab="p2")
}
```



Graphs of the posterior.

```
sigma <- c(2, 1, .5, .25)
par(mfrow=c(2, 2))
for (i in 1:4){
   mycontour(howardprior,
             c(plo, phi, plo, phi),
     c(1 + 3, 1 + 15, 1 + 7, 1 + 5, sigma[i]),
     main=paste("sigma=", as.character(sigma[i])),
     xlab="p1", ylab="p2")
   lines(c(0, 1), c(0, 1))
 }
```




```
s <- simcontour(howardprior, c(plo, phi, plo, phi),
   c(1 + 3, 1 + 15, 1 + 7, 1 + 5, 2), 1000)
sum(s$x > s$y) / 1000
```

```
## [1] 0.012
```

Chapter 5

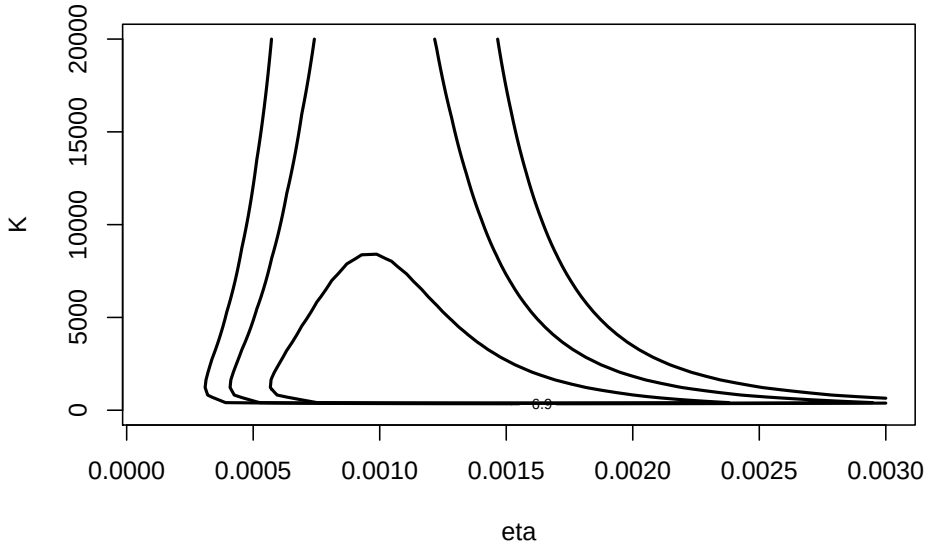
Introduction to Bayesian Computation

```
library(LearnBayes)
```

5.1 A Beta-Binomial Model for Overdispersion

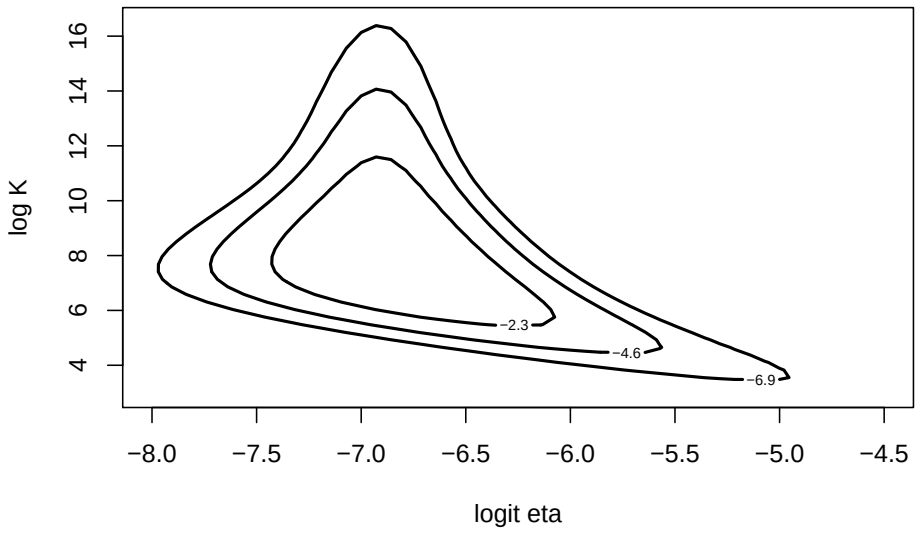
First consider posterior of (η, K).

```
mycontour(betabinexch0,
          c(.0001, .003, 1, 20000),
          cancermortality,
          xlab="eta", ylab="K")
```



Instead look at posterior of $(\log \frac{\eta}{1-\eta}, \log I$.

```
mycontour(betabinexch,
          c(-8, -4.5, 3, 16.5),
          cancermortality,
          xlab="logit eta", ylab="log K")
```

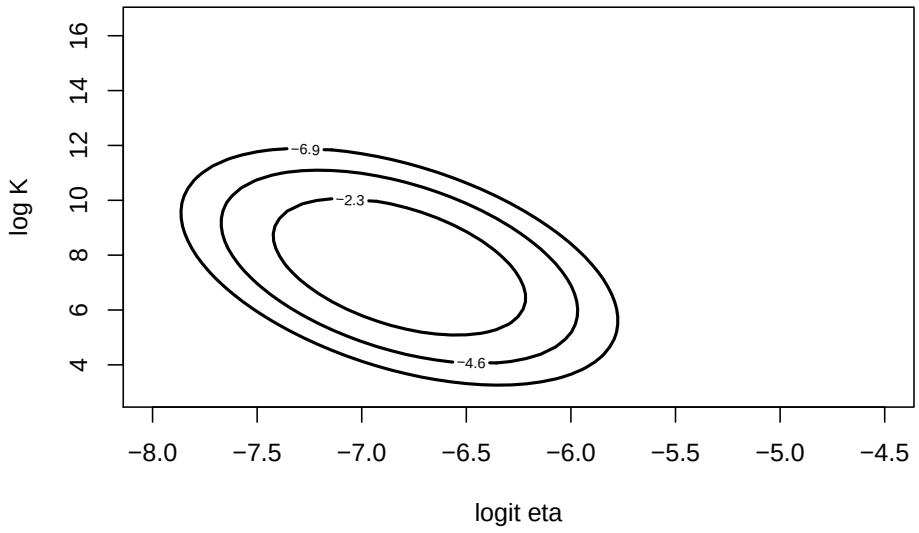


5.2 Approximations Based on Posterior Modes

```
fit <- laplace(betabinexch,
               c(-7, 6),
               cancermortality)
fit
```

```
## $mode
## [1] -6.819793  7.576111
##
## $var
##              [,1]       [,2]
## [1,]  0.07896568 -0.1485087
## [2,] -0.14850874  1.3483208
##
## $int
## [1] -570.7743
##
## $converge
## [1] TRUE
```

```
npar <- list(m=fit$mode, v=fit$var)
mycontour(lbinorm,
          c(-8, -4.5, 3, 16.5),
          npar,
          xlab="logit eta", ylab="log K")
```




```
se <- sqrt(diag(fit$var))
fit$mode - 1.645 * se
```

```
## [1] -7.282052  5.665982
```

```
fit$mode + 1.645 * se
```

```
## [1] -6.357535  9.486239
```

5.3 Monte Carlo Method for Computing Integrals

Illustration of a simple estimate of an integral by Monte Carlo.

```
p <- rbeta(1000, 14.26, 23.19)
est <- mean(p ^ 2)
se <- sd(p ^ 2) / sqrt(1000)
c(est,se)
```

```
## [1] 0.151521812 0.001944763
```

5.4 Rejection Sampling

Using rejection sampling for the overdispersion posterior with a multivariate t proposal density.

```
fit <- laplace(betabinexch,
               c(-7, 6),
               cancermortality)
```

```
betabinT <- function(theta, datapar){
  data <- datapar$data
  tpar <-datapar$par
  d <- betabinexch(theta,data) -
         dmt(theta, mean=c(tpar$m),
             S=tpar$var, df=tpar$df, log=TRUE)
  d
}
```

```
tpar <- list(m=fit$mode, var=2 * fit$var, df=4)
datapar <- list(data=cancermortality, par=tpar)
```

```
start <- c(-6.9, 12.4)
fit1 <- laplace(betabinT, start, datapar)
fit1$mode
```

```
## [1] -6.888963 12.421993
```

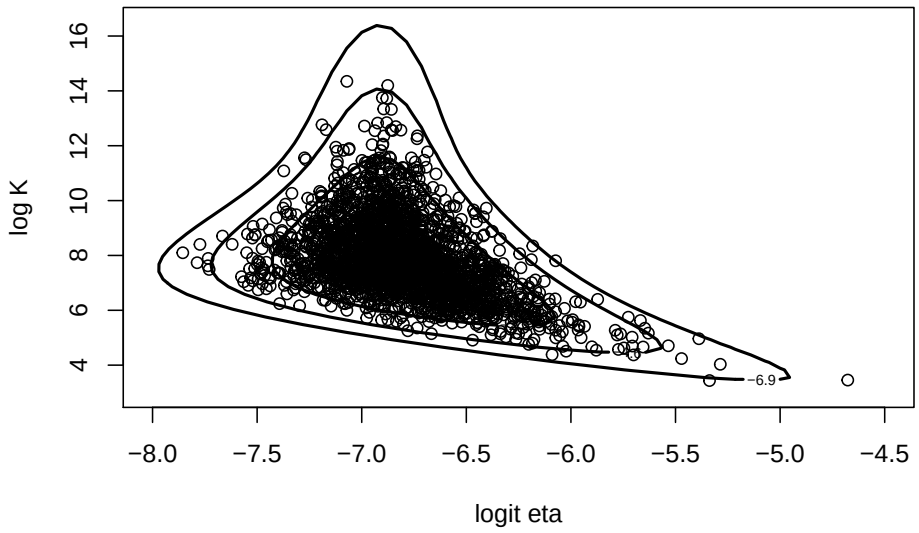
```
betabinT(fit1$mode, datapar)
```

```
## [1] -569.2829
```

```
theta <- rejectsampling(betabinexch,
                        tpar,
                       -569.2813,
                       10000,
                       cancermortality)
dim(theta)
```

```
## [1] 2389    2
```

```
mycontour(betabinexch,
          c(-8, -4.5, 3, 16.5),
          cancermortality,
          xlab="logit eta", ylab="log K")
points(theta[,1],theta[,2])
```



5.5 Importance Sampling

```
fit <- laplace(betabinexch,
               c(-7, 6),
               cancermortality)
```

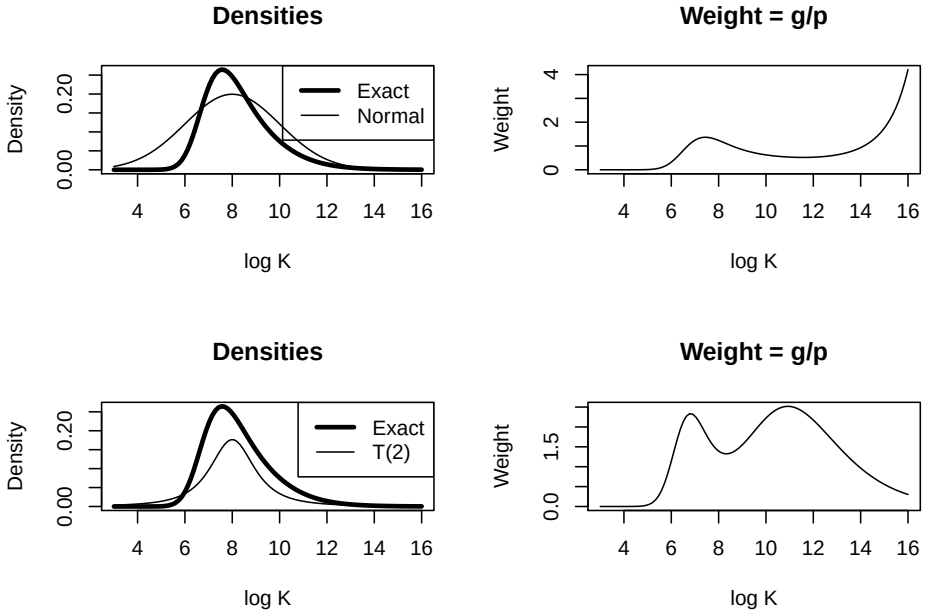
Posterior density of log K\$ conditional on a value of η.

```
betabinexch.cond <- function (log.K, data){
  eta <- exp(-6.818793) / (1 + exp(-6.818793))
  K <- exp(log.K)
  y <- data[, 1]
  n <- data[, 2]
```

```
  N <- length(y)
  logf <- 0 * log.K
  for (j in 1:length(y)){
     logf = logf + lbeta(K * eta + y[j],
                  K * (1 - eta) + n[j] - y[j]) -
                 lbeta(K * eta, K * (1 - eta))
  }
  val <- logf + log.K - 2 * log(1 + K)
  exp(val-max(val))
}
```

Illustrate different choices of importance sampler.

```
I <- integrate(betabinexch.cond, 2, 16,
                cancermortality)
par(mfrow=c(2, 2))
curve(betabinexch.cond(x,
            cancermortality) / I$value,
            from=3, to=16,
            ylab="Density", xlab="log K", lwd=3,              main="Densities")
curve(dnorm(x, 8, 2), add=TRUE)
legend("topright",
       legend=c("Exact", "Normal"),
       lwd=c(3, 1))
curve(betabinexch.cond(x,
            cancermortality) / I$value /
           dnorm(x, 8, 2), from=3, to=16,                 ylab="Weight", xlab="log K",
       main="Weight = g/p")
curve(betabinexch.cond(x,
            cancermortality) /I$value,
          from=3, to=16,
          ylab="Density", xlab="log K",
          lwd=3, main="Densities")
curve(1 / 2 * dt(x - 8, df=2), add=TRUE)
legend("topright", legend=c("Exact", "T(2)"), lwd=c(3, 1))
curve(betabinexch.cond(x,
          cancermortality) / I$value /
        (1 / 2 * dt(x - 8, df=2)),
        from=3, to=16,
        ylab="Weight", xlab="log K",
        main="Weight = g/p")
```




```
tpar <- list(m=fit$mode,
             var=2 * fit$var,
             df=4)
myfunc <- function(theta){
   return(theta[2])
}
s <- impsampling(betabinexch,
                 tpar,
                 myfunc,
                 10000,
                 cancermortality)
cbind(s$est, s$se)
```

```
##          [,1]       [,2]
## [1,] 7.965118 0.01952959
```

5.6 Sampling Importance Resampling

Illustrate using the SIR algorithm for the beta-binomial density with a multivariate t proposal density.

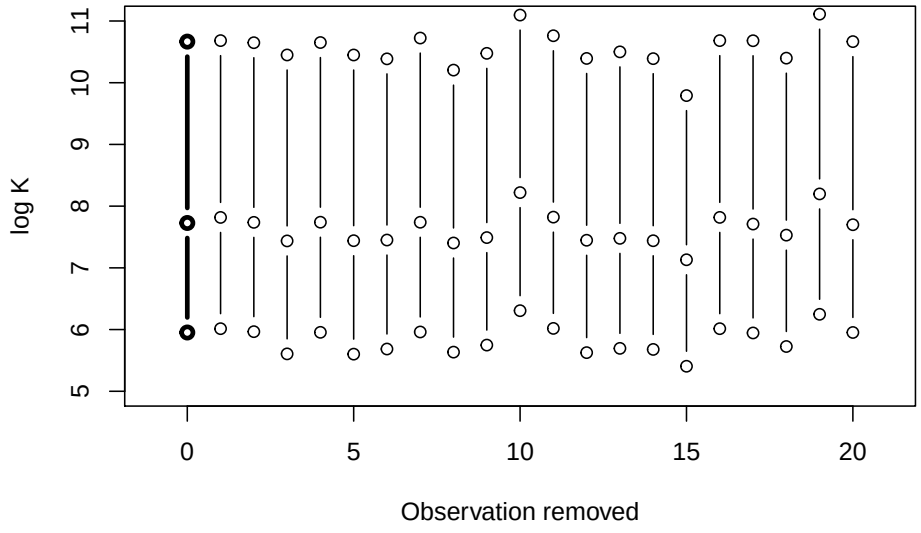
```
fit <- laplace(betabinexch,
               c(-7, 6),
               cancermortality)
```

```
tpar <- list(m=fit$mode,
             var=2 * fit$var, df=4)
```

```
theta.s <- sir(betabinexch,
               tpar, 10000,
               cancermortality)
```

Use SIR to examine the sensitivity of the posterior inference to removal of individual observations.

```
S <- bayes.influence(theta.s, cancermortality)
plot(c(0, 0, 0), S$summary,
     type="b", lwd=3, xlim=c(-1, 21),
     ylim=c(5, 11),
     xlab="Observation removed",  ylab="log K")
for (i in 1:20){
  lines(c(i, i, i), S$summary.obs[i, ], type="b")
}
```



Chapter 6

Markov Chain Monte Carlo Methods

6.1 Introduction to Discrete Markov Chains

Illustration of sampling from a random walk distribution.

```
P <- matrix(c(.5, .5, 0, 0, 0, 0,
           .25, .5, .25, 0, 0, 0,
           0, .25, .5, .25, 0, 0,
           0, 0, .25, .5, .25, 0,
           0, 0, 0, .25, .5, .25,
           0, 0, 0, 0, .5, .5),
           nrow=6, ncol=6, byrow=TRUE)
 P
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,] 0.50 0.50 0.00 0.00 0.00 0.00
## [2,] 0.25 0.50 0.25 0.00 0.00 0.00
## [3,] 0.00 0.25 0.50 0.25 0.00 0.00
## [4,] 0.00 0.00 0.25 0.50 0.25 0.00
## [5,] 0.00 0.00 0.00 0.25 0.50 0.25
## [6,] 0.00 0.00 0.00 0.00 0.50 0.50
```

```
s <- array(0, c(50000, 1))
s[1] <- 3
for (j in 2:50000){
   s[j] <- sample(1:6, size=1, prob=P[s[j - 1],])
}
```

```
m <- c(500, 2000, 8000, 50000)
for (i in 1:4){
   print(table(s[1:m[i]]) / m[i])
}
```

```
##
##     1     2     3     4     5     6
## 0.138 0.158 0.142 0.194 0.236 0.132
##
##      1      2      3      4      5      6
## 0.1010 0.1895 0.1810 0.1905 0.2080 0.1300
##
##        1        2        3        4        5        6
## 0.111250 0.209375 0.195000 0.190625 0.186625 0.107125
##
##       1       2       3       4       5       6
## 0.10062 0.19684 0.20054 0.20030 0.19934 0.10236
```

```
w <- matrix(c(.1, .2, .2, .2, .2, .1),
            nrow=1, ncol=6)
w %*% P
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]  0.1  0.2  0.2  0.2  0.2  0.1
```

6.2 Learning about a Normal Population from Grouped Data

Have normally distributed data where the data is observed in grouped form. Consider the posterior of $(\mu, \log \sigma)$.

```
d <- list(int.lo=c(-Inf, seq(66, 74, by=2)),
        int.hi=c(seq(66, 74, by=2), Inf),
        f=c(14, 30, 49, 70, 33, 15))
```

```
y <- c(rep(65,14), rep(67,30), rep(69,49),
       rep(71,70), rep(73,33), rep(75,15))
mean(y)
```

```
## [1] 70.16588
```

```
log(sd(y))
```

```
## [1] 0.9504117
```

First obtain normal approximation to posterior.

```
start <- c(70, 1)
fit <- laplace(groupeddatapost, start, d)
fit
```

```
## $mode
## [1] 70.169880  0.973644
##
## $var
##               [,1]         [,2]
## [1,] 3.534713e-02 3.520776e-05
## [2,] 3.520776e-05 3.146470e-03
##
## $int
## [1] -350.6305
##
## $converge
## [1] TRUE
```

Now use a Metropolis (random walk) MCMC algorithm.

```
modal.sds <- sqrt(diag(fit$var))
proposal <- list(var=fit$var, scale=2)
fit2 <- rwmetrop(groupeddatapost,
                 proposal,
                 start,
                 10000, d)
```

```
fit2$accept
```

```
## [1] 0.3011
```

```
post.means <- apply(fit2$par, 2, mean)
post.sds <- apply(fit2$par, 2, sd)
cbind(c(fit$mode), modal.sds)
```

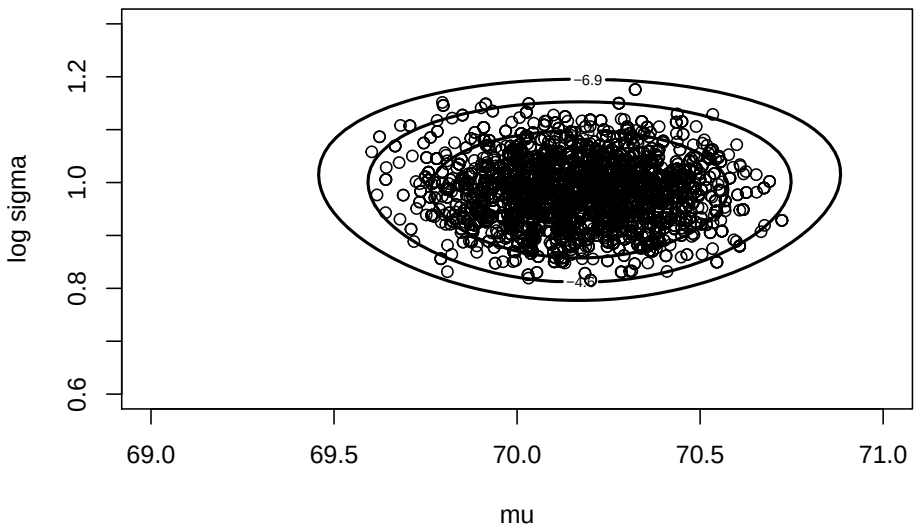
```
##                  modal.sds
## [1,] 70.169880 0.18800834
## [2,]  0.973644 0.05609341
```

```
cbind(post.means, post.sds)
```

```
##      post.means   post.sds
## [1,] 70.1683845 0.18297635
## [2,]  0.9803531 0.05545176
```

```
mycontour(groupeddatapost,
          c(69, 71, .6, 1.3), d,
       xlab="mu",ylab="log sigma")
points(fit2$par[5001:10000, 1],
```

```
       fit2$par[5001:10000, 2])
```



6.3 Example of Output Analysis

Illustrate MCMC diagnositics for different Metropolis chains with different proposal widths.

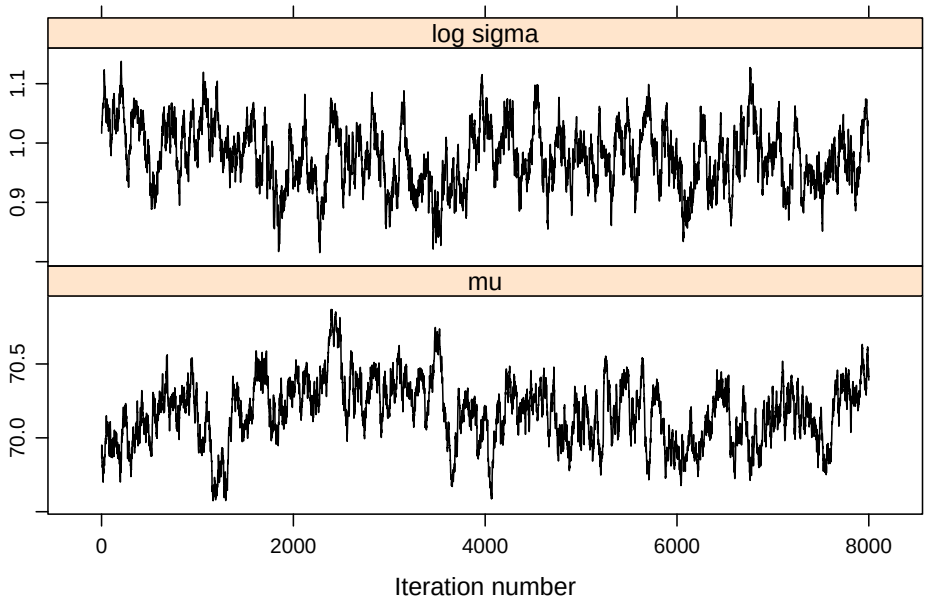
```
d <- list(int.lo=c(-Inf, seq(66, 74, by=2)),
        int.hi=c(seq(66, 74, by=2), Inf),
        f=c(14, 30, 49, 70, 33, 15))
```

```
library(coda)
library(lattice)
```

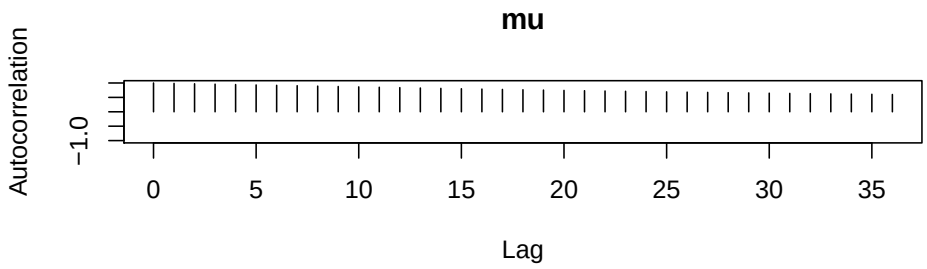
```
start <- c(70,1)
 fit <- laplace(groupeddatapost, start, d)
```

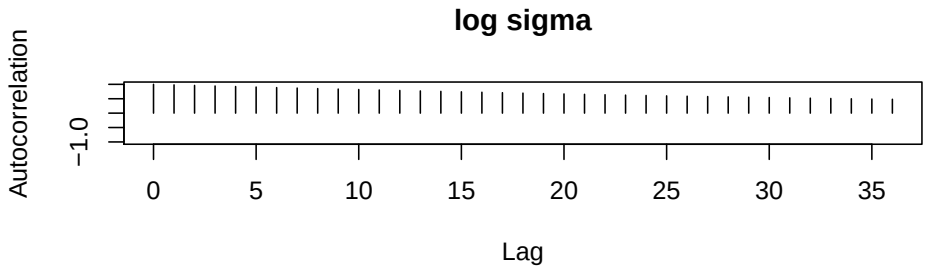
```
start <- c(65,1)
 proposal <- list(var=fit$var, scale=0.2)
 bayesfit <- rwmetrop(groupeddatapost,
                      proposal,
                      start,
                      10000, d)
```

```
dimnames(bayesfit$par)[[2]] <- c("mu", "log sigma")
 xyplot(mcmc(bayesfit$par[-c(1:2000), ]),
        col="black")
```




```
par(mfrow=c(2, 1))
 autocorr.plot(mcmc(bayesfit$par[-c(1:2000), ]),
               auto.layout=FALSE)
```






```
 summary(mcmc(bayesfit$par[-c(1:2000), ]))
```

```
##
```

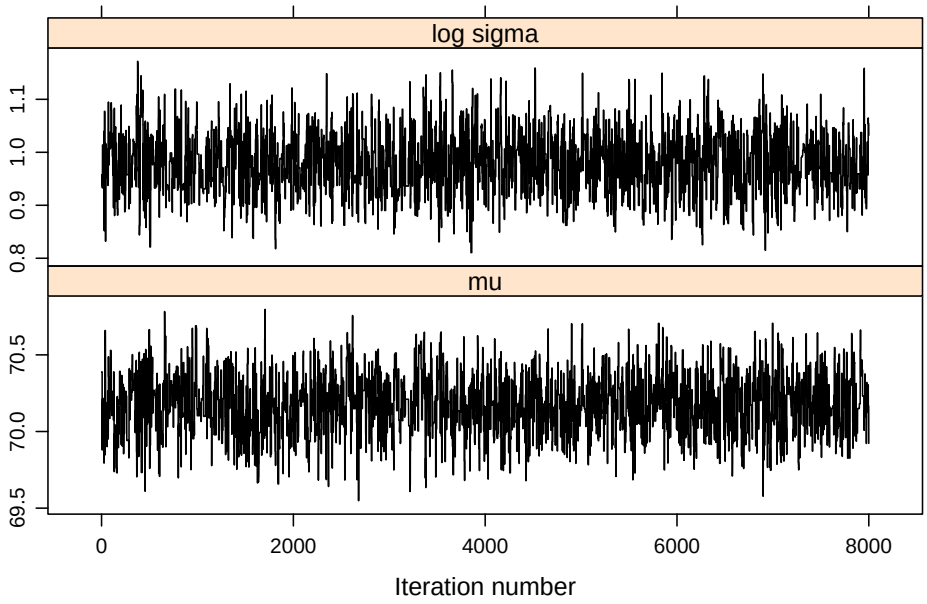
```
## Iterations = 1:8000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 8000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean      SD  Naive SE Time-series SE
## mu        70.1703 0.21342 0.0023861       0.028804
## log sigma  0.9774 0.05308 0.0005934       0.005875
##
## 2. Quantiles for each variable:
##
##              2.5%     25%     50%    75%  97.5%
## mu        69.7478 70.0242 70.1792 70.316 70.589
## log sigma  0.8756  0.9395  0.9761  1.017  1.076
```

```
batchSE(mcmc(bayesfit$par[-c(1:2000), ]),
        batchSize=50)
```

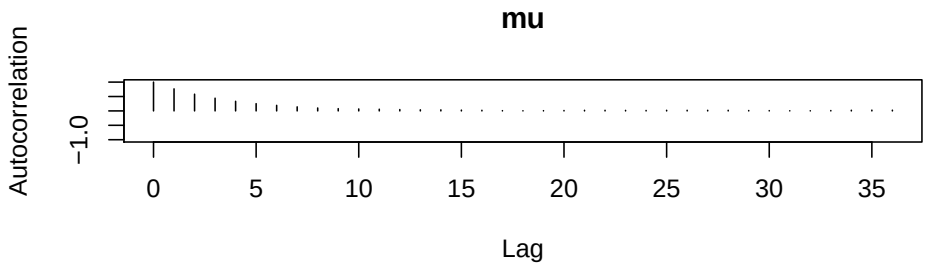
```
##          mu   log sigma
## 0.015097003 0.003629486
```

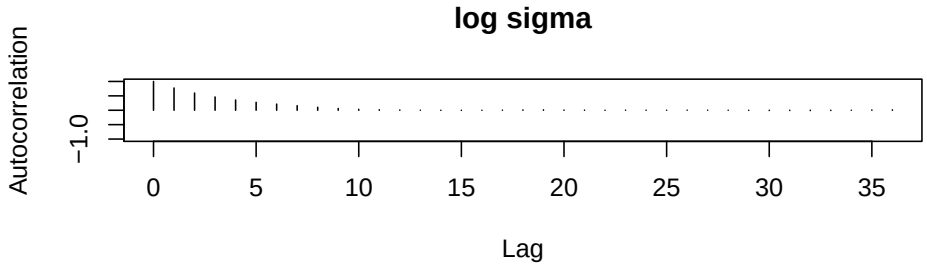
```
start <- c(70,1)
proposal <- list(var=fit$var, scale=2.0)
bayesfit <- rwmetrop(groupeddatapost,
                     proposal,
                     start,
                     10000, d)
```

```
dimnames(bayesfit$par)[[2]] <- c("mu", "log sigma")
sim.parameters <- mcmc(bayesfit$par[-c(1:2000), ])
xyplot(mcmc(bayesfit$par[-c(1:2000), ]),
       col="black")
```




```
par(mfrow=c(2,1))
 autocorr.plot(sim.parameters,auto.layout=FALSE)
```






```
 summary(sim.parameters)

##
## Iterations = 1:8000
```

```
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 8000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean      SD  Naive SE Time-series SE
## mu        70.1768 0.19412 0.0021703       0.005967
## log sigma  0.9795 0.05732 0.0006409       0.001795
##
## 2. Quantiles for each variable:
##
##              2.5%     25%     50%    75%  97.5%
## mu        69.7923 70.0437 70.1745 70.309 70.556
## log sigma  0.8677  0.9398  0.9804  1.019  1.088
```

```
batchSE(sim.parameters, batchSize=50)
```

```
##          mu   log sigma
## 0.006082428 0.001632771
```

6.4 Modeling Data with Cauchy Errors

Assuming data that is sampled from a Cauchy density with a noninformative prior placed on the location and scale parameters.

```
mean(darwin$difference)
```

```
## [1] 21.66667
```

```
log(sd(darwin$difference))
```

```
## [1] 3.65253
```

First illustrate normal approximation.

```
laplace(cauchyerrorpost,
        c(21.6, 3.6),
        darwin$difference)
```

```
## $mode
## [1] 24.701745  2.772619
##
## $var
##            [,1]      [,2]
## [1,] 34.9600525 0.3672899
## [2,]  0.3672899 0.1378279
##
```

```
## $int
## [1] -73.2404
##
## $converge
## [1] TRUE
```

```
laplace(cauchyerrorpost,
        .1 * c(21.6,3.6),
        darwin$difference)$mode
```

```
## [1] 24.698151  2.772345
```

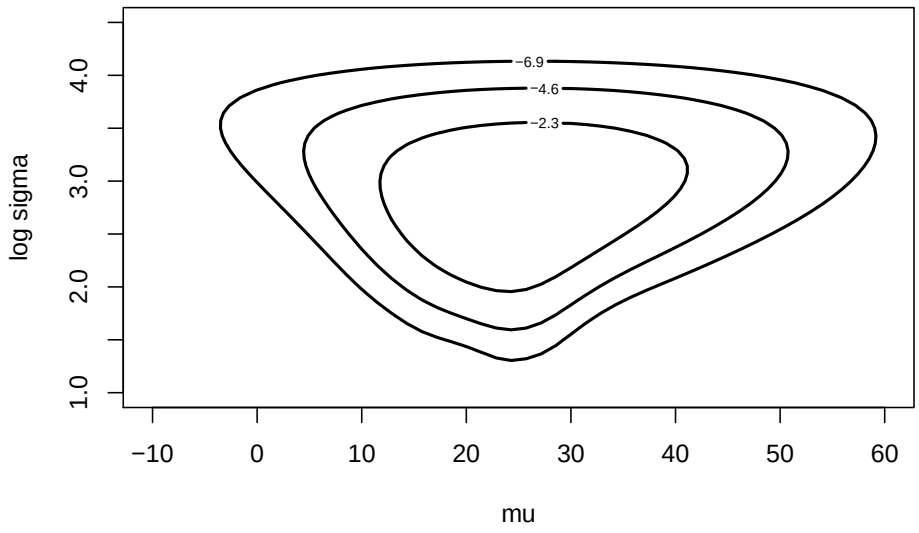
```
c(24.7 - 4 * sqrt(34.96), 24.7 + 4 * sqrt(34.96))
```

```
## [1]  1.049207 48.350793
```

```
c(2.77 - 4 * sqrt(.138), 2.77 + 4 * sqrt(.138))
```

```
## [1] 1.284066 4.255934
```

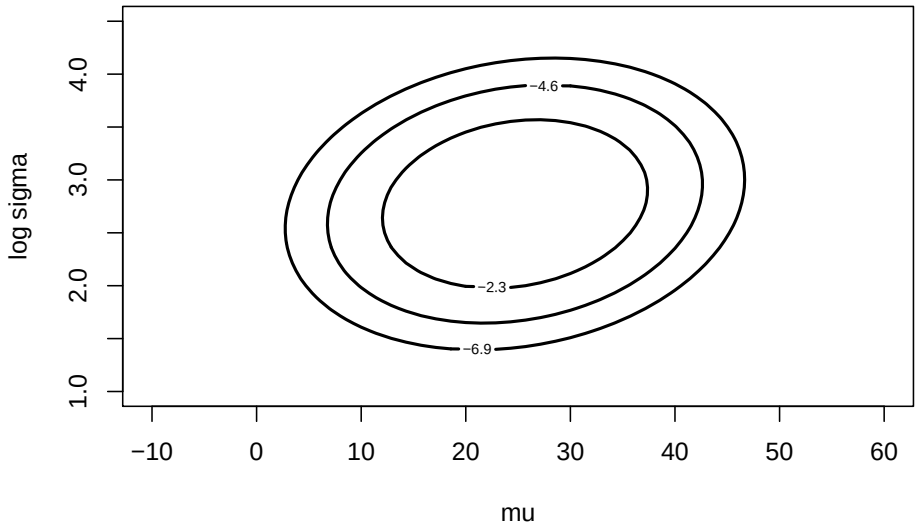
```
mycontour(cauchyerrorpost,
          c(-10, 60, 1, 4.5),
          darwin$difference,
          xlab="mu", ylab="log sigma")
```




```
fitlaplace <- laplace(cauchyerrorpost,
                      c(21.6, 3.6),
                      darwin$difference)
```

```
mycontour(lbinorm,
          c(-10, 60, 1, 4.5),
          list(m=fitlaplace$mode,
```

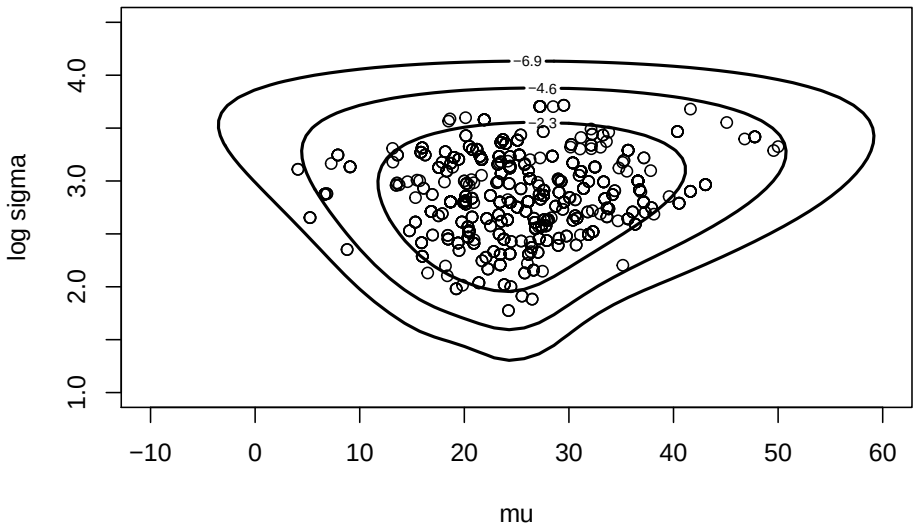
```
                v=fitlaplace$var),
          xlab="mu",ylab="log sigma")
```



Next illustrate random walk Metropolis.

```
proposal <- list(var=fitlaplace$var, scale=2.5)
start <- c(20, 3)
m <- 1000
s <- rwmetrop(cauchyerrorpost, proposal,
              start, m, darwin$difference)
```

```
mycontour(cauchyerrorpost,
          c(-10, 60, 1, 4.5),
          darwin$difference,
          xlab="mu", ylab="log sigma")
 points(s$par[,1], s$par[,2])
```




```
fitgrid <- simcontour(cauchyerrorpost,
                      c(-10,60,1,4.5),
                      darwin$difference,
                      50000)
```

```
proposal <- list(var=fitlaplace$var,
                 scale=2.5)
                 start=c(20, 3)
fitrw=rwmetrop(cauchyerrorpost,
               proposal,
               start,
               50000,
               darwin$difference)
```

Illustrate metropolis-hastings independence chain.

```
proposal2 <- list(var=fitlaplace$var,
                  mu=t(fitlaplace$mode))
fitindep <- indepmetrop(cauchyerrorpost,
                        proposal2,
                        start,
                        50000,
                        darwin$difference)
```

Illustrate metropolis-within-Gibbs.

```
fitgibbs <- gibbs(cauchyerrorpost,
                  start,
                  50000,
                  c(12,.75),
                  darwin$difference)
```

```
apply(fitrw$par,2,mean)
```

```
## [1] 25.461642  2.838586
```

```
apply(fitrw$par,2,sd)
```

```
## [1] 6.9419258 0.3693491
```

6.5 Analysis of the Stanford Heart Transplant Data

Using a Pareto model to analyze heart transplant data.

Laplace fit.

```
start <- c(0, 3, -1)
laplacefit <- laplace(transplantpost,
                      start, stanfordheart)
laplacefit
```

```
## $mode
## [1] -0.09210954  3.38385249 -0.72334008
##
## $var
##               [,1]         [,2]        [,3]
## [1,]  0.172788525 -0.009282308 -0.04995160
## [2,] -0.009282308  0.214737054  0.09301323
## [3,] -0.049951602  0.093013230  0.06891796
##
## $int
## [1] -376.2504
##
## $converge
## [1] TRUE
```

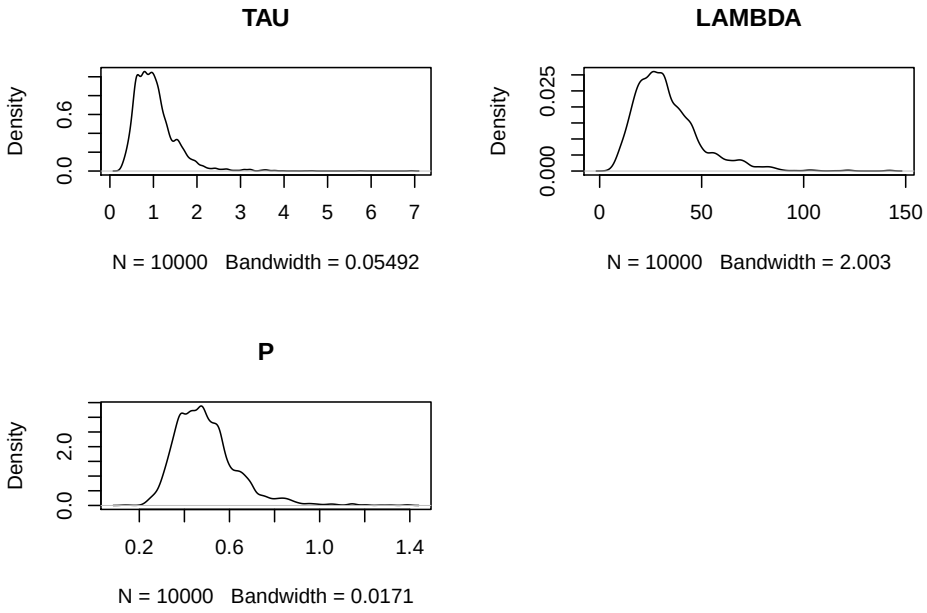
Random walk metropolis.

```
proposal <- list(var=laplacefit$var, scale=2)
s <- rwmetrop(transplantpost,
              proposal,
              start, 10000, stanfordheart)
s$accept
```

```
## [1] 0.1878
```

```
par(mfrow=c(2,2))
tau <- exp(s$par[,1])
plot(density(tau), main="TAU")
```

```
lambda <- exp(s$par[,2])
plot(density(lambda), main="LAMBDA")
p <- exp(s$par[,3])
plot(density(p), main="P")
```



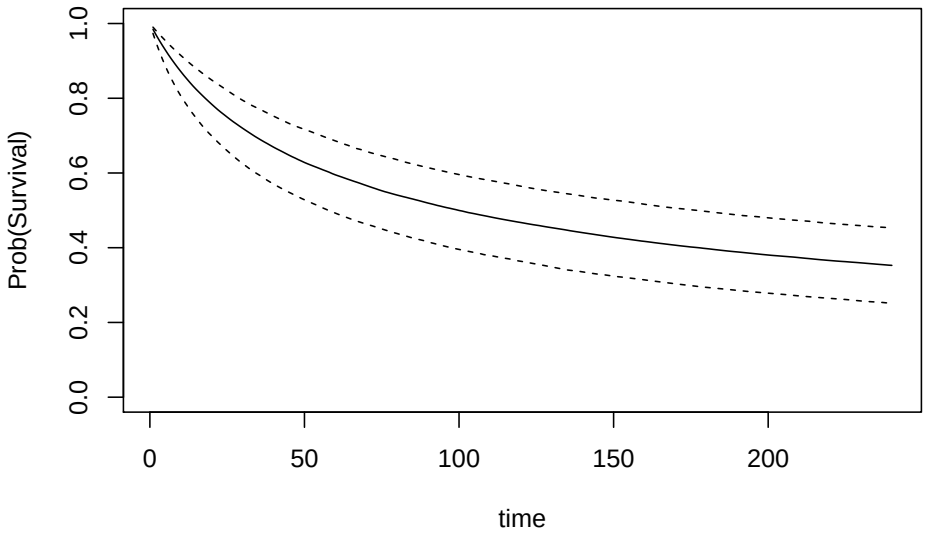

```
apply(exp(s$par), 2, quantile, c(.05, .5, .95))
```

```
##          [,1]     [,2]      [,3]
## 5%  0.4816982 13.52028 0.3185855
## 50% 0.9500635 30.09466 0.4760481
## 95% 1.8746643 65.04539 0.7455402
```

```
par(mfrow=c(1, 1))
t <- seq(1, 240)
p5 <- 0*t
p50 <- 0 * t
p95 <- 0 * t
for (j in 1:240){
   S <- (lambda / (lambda + t[j])) ^ p
   q <- quantile(S, c(.05, .5, .95))
   p5[j] <- q[1]
   p50[j] <- q[2]
   p95[j] <- q[3]
}
```

Estimating a patient's survival curve.

```
plot(t, p50, type="l",
     ylim=c(0,1),
     ylab="Prob(Survival)",
     xlab="time")
 lines(t, p5, lty=2)
 lines(t, p95, lty=2)
```



Chapter 7

Hierarchical Modeling

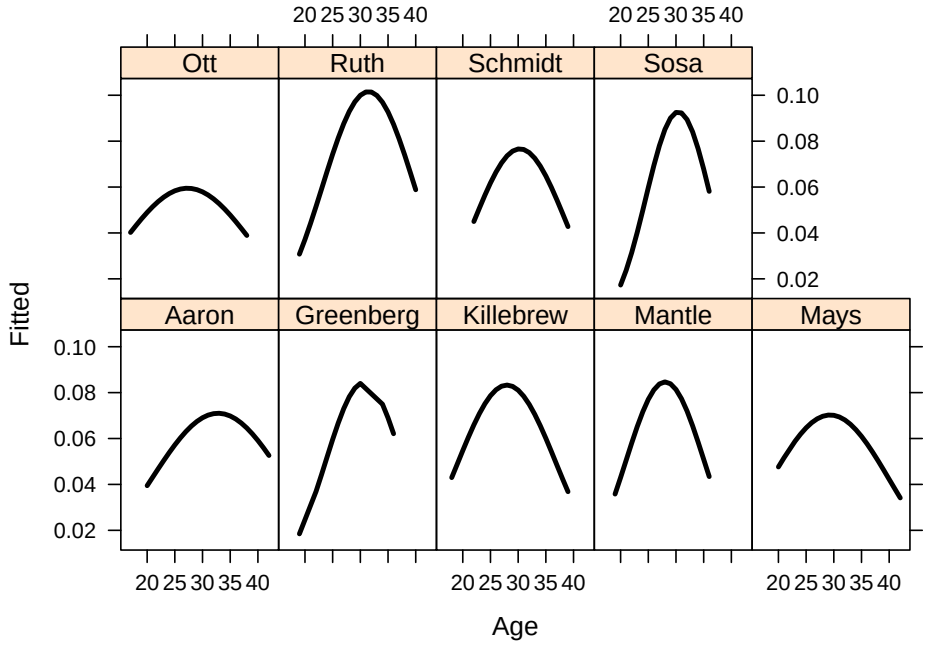
7.1 Introduction to Hierarchical Modeling

```
library(LearnBayes)
library(lattice)
```

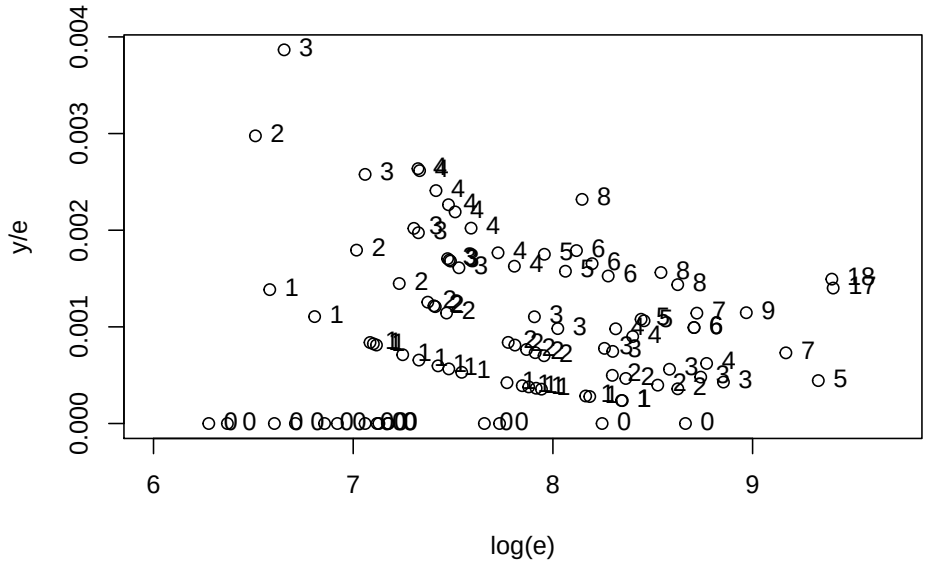
Fit logistic model for home run data for a particular player

```
logistic.fit <- function(player){
  d <- subset(sluggerdata, Player==player)
  x <- d$Age
  x2 <- d$Age^2
  response <- cbind(d$HR, d$AB - d$HR)
  list(Age=x,
       p=glm(response ~ x + x2,
             family=binomial)$fitted)
}
```

```
names <- unique(sluggerdata$Player)
newdata <- NULL
for (j in 1:9){
  fit <-logistic.fit(as.character(names[j]))
  newdata <- rbind(newdata,
            data.frame(as.character(names[j]),
                       fit$Age, fit$p))
}
names(newdata) <- c("Player", "Age", "Fitted")
xyplot(Fitted ~ Age | Player,
       data=newdata,
       type="l", lwd=3, col="black")
```




```
with(hearttransplants,
      plot(log(e), y / e, xlim=c(6, 9.7),
           xlab="log(e)", ylab="y/e"))
 with(hearttransplants,
      text(log(e), y / e,
           labels=as.character(y), pos=4))
```



7.3 Equal Mortality Rates?

Using posterior predictive checks to see if equal mortality rate model is appropriate.

```
with(hearttransplants, sum(y))
```

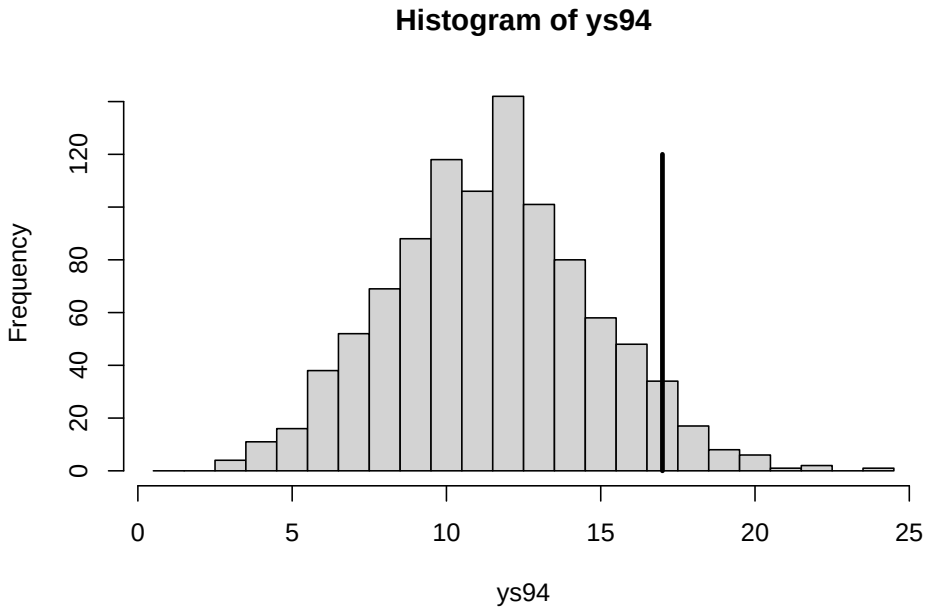
```
## [1] 277
```

```
with(hearttransplants, sum(e))
```

```
## [1] 294681
```

```
lambda <- rgamma(1000, shape=277, rate=294681)
ys94 <- with(hearttransplants,
      rpois(1000, e[94] * lambda))
```

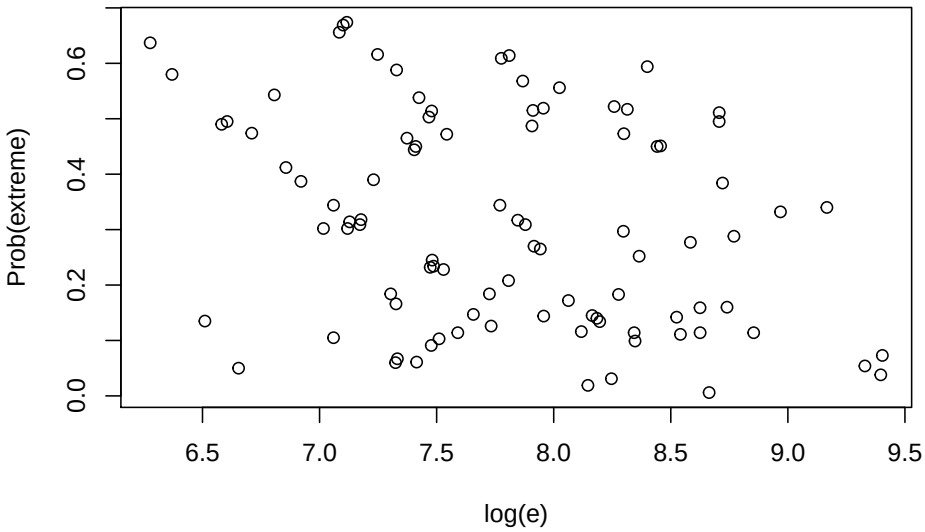
```
hist(ys94, breaks=seq(0.5, max(ys94) + 0.5))
with(hearttransplants,
      lines(c(y[94], y[94]), c(0, 120), lwd=3))
```



Find posterior predictive distribution of each observation with its posterior predictive distribution.

```
lambda <- rgamma(1000, shape=277, rate=294681)
prob.out <- function(i){
   ysi <- with(hearttransplants,
            rpois(1000, e[i] * lambda))
   pleft <- with(hearttransplants,
              sum(ysi <= y[i]) / 1000)
   pright <- with(hearttransplants,
               sum(ysi >= y[i]) / 1000)
   min(pleft, pright)
 }
pout <- sapply(1:94, prob.out)
```

```
with(hearttransplants,
     plot(log(e), pout, ylab="Prob(extreme)"))
```

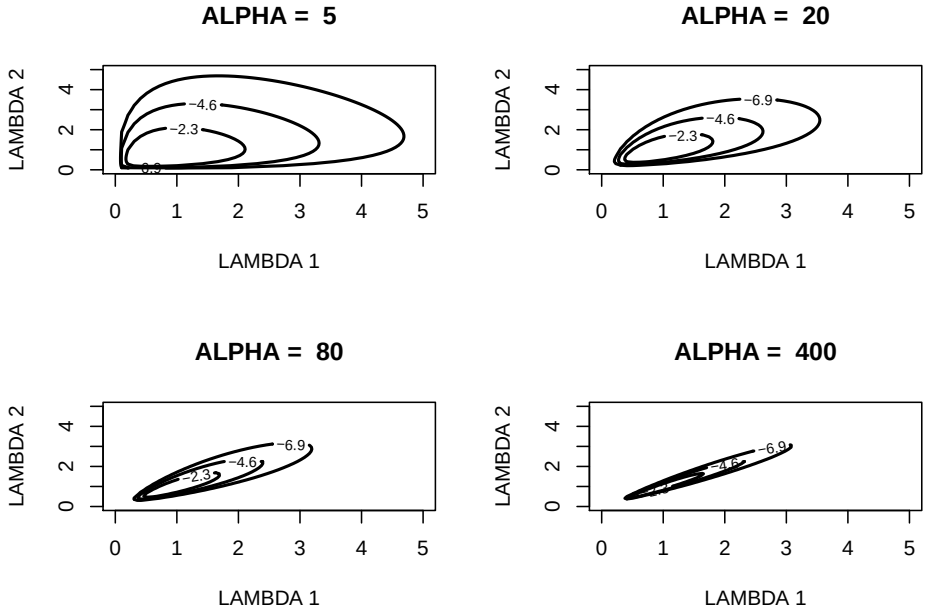


7.4 Modeling a Prior Belief of Exchangeability

Graph of two-stage prior to model a belief in exchangeability of the Poisson rates.

```
pgexchprior <- function(lambda, pars){
alpha <- pars[1]
a <- pars[2]
b <- pars[3]
(alpha - 1) * log(prod(lambda)) -
  (2 * alpha + a) * log(alpha * sum(lambda) + b)
}
```

```
alpha <- c(5, 20, 80, 400)
par(mfrow=c(2, 2))
for (j in 1:4){
    mycontour(pgexchprior,
              c(.001, 5, .001, 5),
              c(alpha[j], 10, 10),
          main=paste("ALPHA = ",alpha[j]),
         xlab="LAMBDA 1", ylab="LAMBDA 2")
}
```



7.5 Simulating from the Posterior

Representing posterior as $[\mu, \alpha]$ $[\{\lambda_j\}|\mu, \alpha]$.

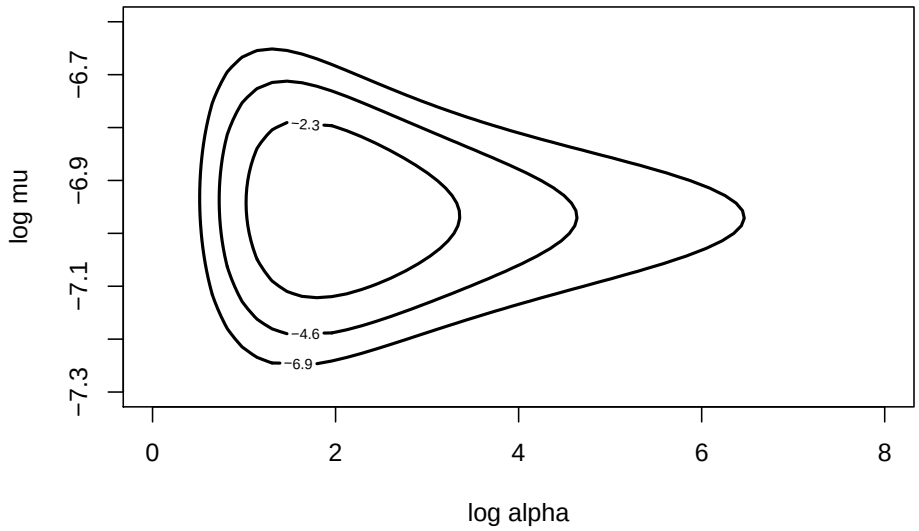
Focus on posterior of $[\mu, \alpha]$:

```
datapar <- list(data = hearttransplants, z0 = 0.53)
start <- c(2, -7)
fit <- laplace(poissgamexch, start, datapar)
 fit
```

```
## $mode
## [1]  1.883954 -6.955446
##
## $var
##               [,1]         [,2]
## [1,]  0.233694921 -0.003086655
## [2,] -0.003086655  0.005866020
##
## $int
## [1] -2208.503
##
## $converge
## [1] TRUE
```

```
par(mfrow = c(1, 1))
mycontour(poissgamexch, c(0, 8, -7.3, -6.6),
```

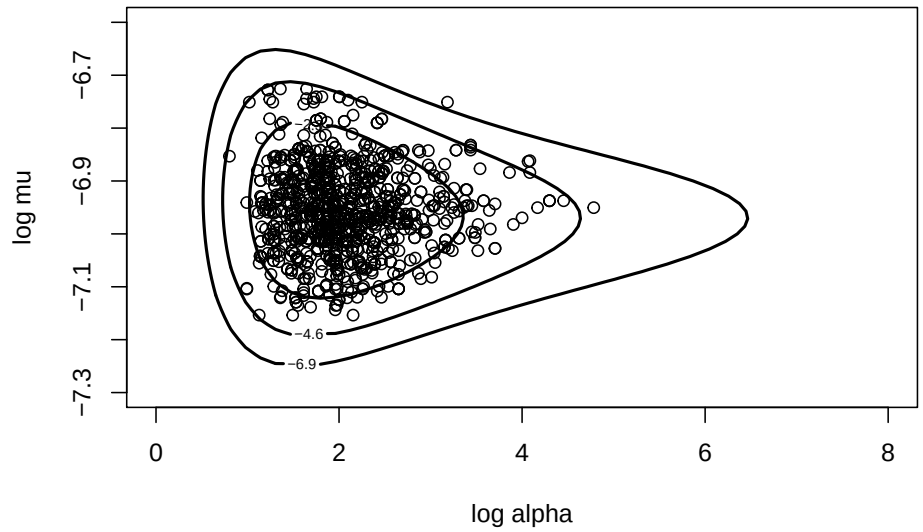
```
          datapar,
          xlab="log alpha", ylab="log mu")
```



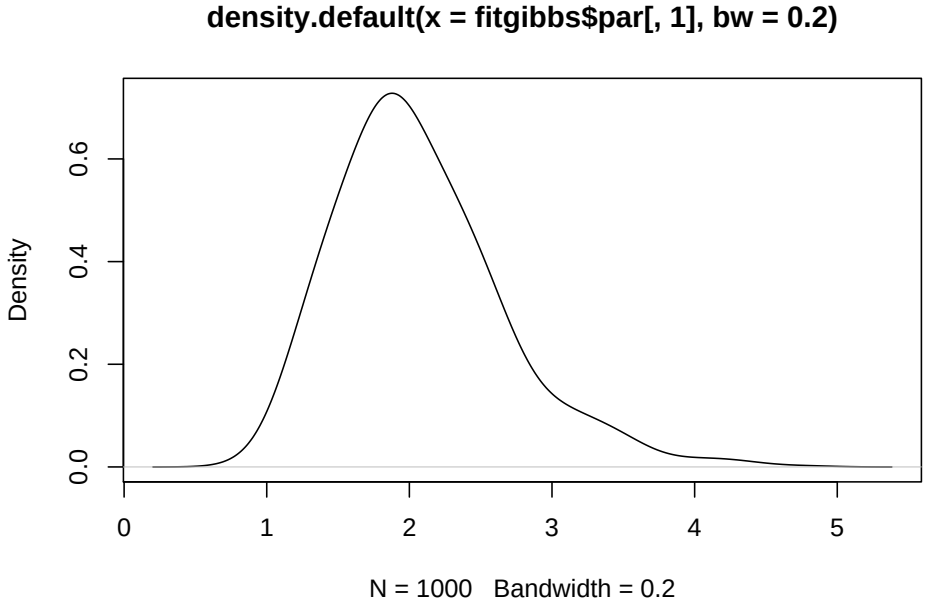

```
start <- c(4, -7)
fitgibbs <- gibbs(poissgamexch,
                  start, 1000,
                  c(1, .15), datapar)
fitgibbs$accept
```

```
##       [,1]  [,2]
## [1,] 0.502 0.476
```

```
mycontour(poissgamexch,
          c(0, 8, -7.3, -6.6),
          datapar,
          xlab="log alpha", ylab="log mu")
points(fitgibbs$par[, 1], fitgibbs$par[, 2])
```



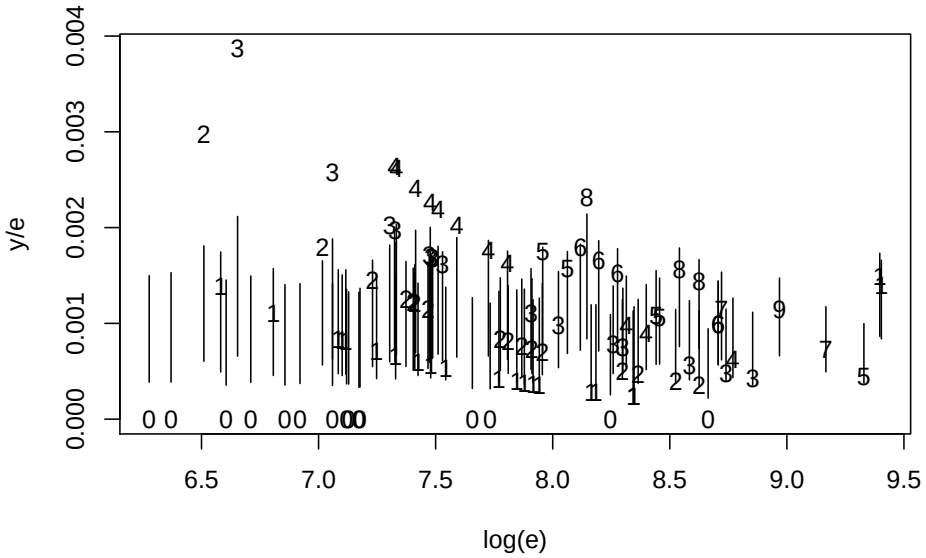

```
plot(density(fitgibbs$par[, 1], bw = 0.2))
```



Posterior of rates:

```
alpha <- exp(fitgibbs$par[, 1])
mu <- exp(fitgibbs$par[, 2])
lam1 <- rgamma(1000, y[1] + alpha,
               hearttransplants$e[1] + alpha / mu)
alpha <- exp(fitgibbs$par[, 1])
mu <- exp(fitgibbs$par[, 2])
```

```
with(hearttransplants,
     plot(log(e), y/e, pch = as.character(y)))
 for (i in 1:94) {
     lami <- with(hearttransplants,
           rgamma(1000, y[i] + alpha,
               e[i] + alpha/mu))
     probint <- quantile(lami, c(0.05, 0.95))
     with(hearttransplants,
          lines(log(e[i]) * c(1, 1), probint))
 }
```



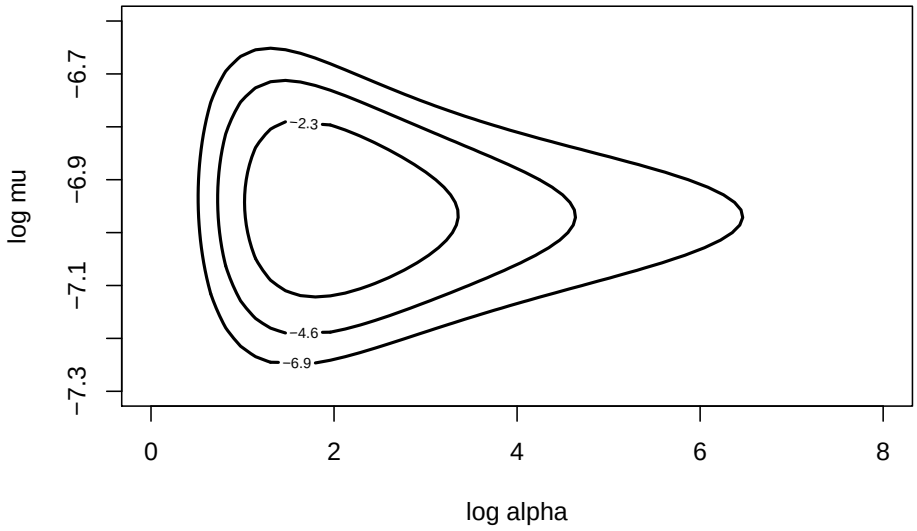
7.6 Posterior Inferences

```
datapar <- list(data = hearttransplants, z0 = 0.53)
start <- c(2, -7)
fit <- laplace(poissgamexch, start, datapar)
fit
```

```
## $mode
## [1]  1.883954 -6.955446
##
## $var
##               [,1]         [,2]
## [1,]  0.233694921 -0.003086655
## [2,] -0.003086655  0.005866020
```

```
##
## $int
## [1] -2208.503
##
## $converge
## [1] TRUE
```

```
par(mfrow = c(1, 1))
mycontour(poissgamexch,
          c(0, 8, -7.3, -6.6), datapar,
          xlab="log alpha",ylab="log mu")
```



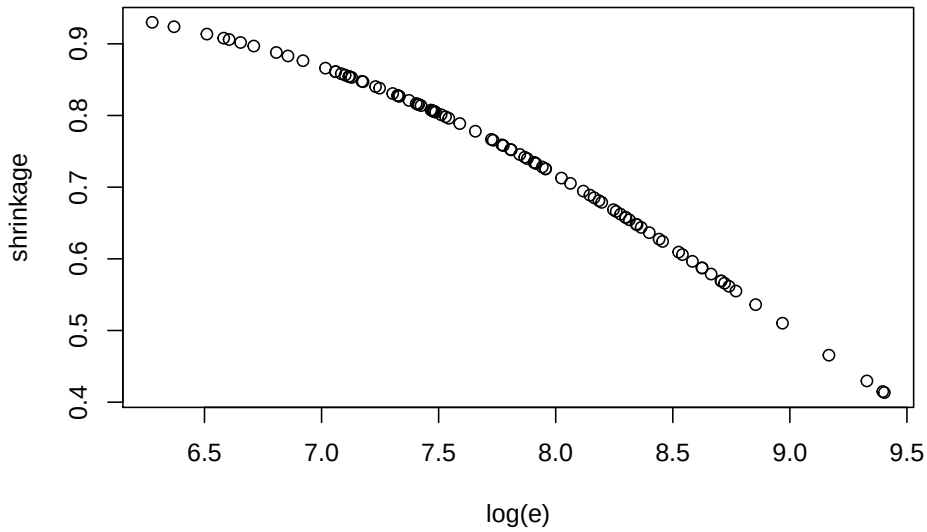

```
start <- c(4, -7)
fitgibbs <- gibbs(poissgamexch,
                  start, 1000,
                  c(1,.15), datapar)
```

```
alpha <- exp(fitgibbs$par[, 1])
mu <- exp(fitgibbs$par[, 2])
```

Look at posteriors of shrinkages.

```
shrink <-function(i)
with(hearttransplants,
       mean(alpha / (alpha + e[i] * mu)))
shrinkage=sapply(1:94, shrink)
```

```
with(hearttransplants,
     plot(log(e), shrinkage))
```



Comparing hospitals.

```
mrate <- function(i){
   with(hearttransplants,
       mean(rgamma(1000, y[i] + alpha,
                   e[i] + alpha/mu)))
}
hospital <- 1:94
meanrate <- sapply(hospital,mrate)
hospital[meanrate == min(meanrate)]
```

```
## [1] 85
```

```
sim.lambda <- function(i) {
  with(hearttransplants,
       rgamma(1000, y[i] + alpha,
              e[i] + alpha / mu))
}
LAM <- sapply(1:94, sim.lambda)
```

```
compare.rates <- function(x) {
  nc <- NCOL(x)
  ij <- as.matrix(expand.grid(1:nc, 1:nc))
  m <- as.matrix(x[,ij[,1]] > x[,ij[,2]])
  matrix(colMeans(m), nc, nc, byrow = TRUE)
}
```

```
better <- compare.rates(LAM)
```

```
better[1:24, 85]
```

```
##  [1] 0.195 0.197 0.095 0.124 0.141 0.231 0.214 0.168 0.079 0.198 0.197 0.162
## [13] 0.198 0.098 0.069 0.209 0.231 0.095 0.265 0.153 0.140 0.154 0.051 0.067
```

7.7 Bayesian Sensitivity Analysis

Explore sensitivity of inference with respect to the choice of z_0 in prior.

```
datapar <- list(data = hearttransplants,
                z0 = 0.53)
```

```
start <- c(4, -7)
fitgibbs <-gibbs(poissgamexch,
                 start, 1000,
                 c(1,.15), datapar)
```

```
sir.old.new <- function(theta, prior, prior.new){
  log.g <- log(prior(theta))
  log.g.new <- log(prior.new(theta))
  wt <- exp(log.g.new - log.g -
              max(log.g.new - log.g))
  probs <- wt / sum(wt)
  n <- length(probs)
  indices <- sample(1:n, size=n,
                    prob=probs, replace=TRUE)
  theta[indices]
}
```

```
prior <- function(theta){
  0.53 * exp(theta) / (exp(theta) + 0.53) ^ 2
}
prior.new <- function(theta){
  5 * exp(theta) / (exp(theta) + 5) ^ 2
}
```

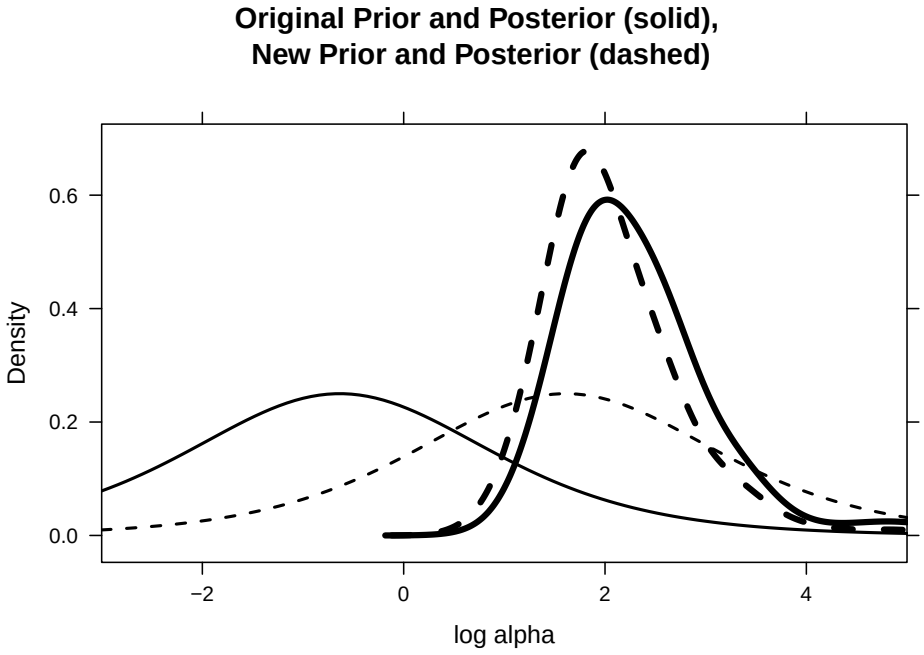
```
log.alpha <- fitgibbs$par[, 1]
log.alpha.new <- sir.old.new(log.alpha,
                             prior, prior.new)
```

```
library(lattice)
```

```
draw.graph <- function(){
  LOG.ALPHA <- data.frame("prior", log.alpha)
  names(LOG.ALPHA) <- c("Prior", "log.alpha")
  LOG.ALPHA.NEW <- data.frame("new.prior",
                              log.alpha.new)
  names(LOG.ALPHA.NEW) <- c("Prior","log.alpha")
  D <- densityplot(~ log.alpha,
```

```
            group=Prior,
            data = rbind(LOG.ALPHA, LOG.ALPHA.NEW),
            plot.points=FALSE,
        main="Original Prior and Posterior (solid), \nNew Prior and Posterior (dashed)",
        lwd=4, adjust=2, lty=c(1,2),
        xlab="log alpha",xlim=c(-3,5),col="black")
    update(D, panel=function(...){
      panel.curve(prior(x), lty=1, lwd=2,
                  col="black")
      panel.curve(prior.new(x), lty=2, lwd=2,
                  col="black")
    panel.densityplot(...)
})}
```

```
draw.graph()
```



7.8 Posterior Predictive Model Checking

Study predictive distributions of observations.

```
datapar <- list(data = hearttransplants, z0 = 0.53)
```

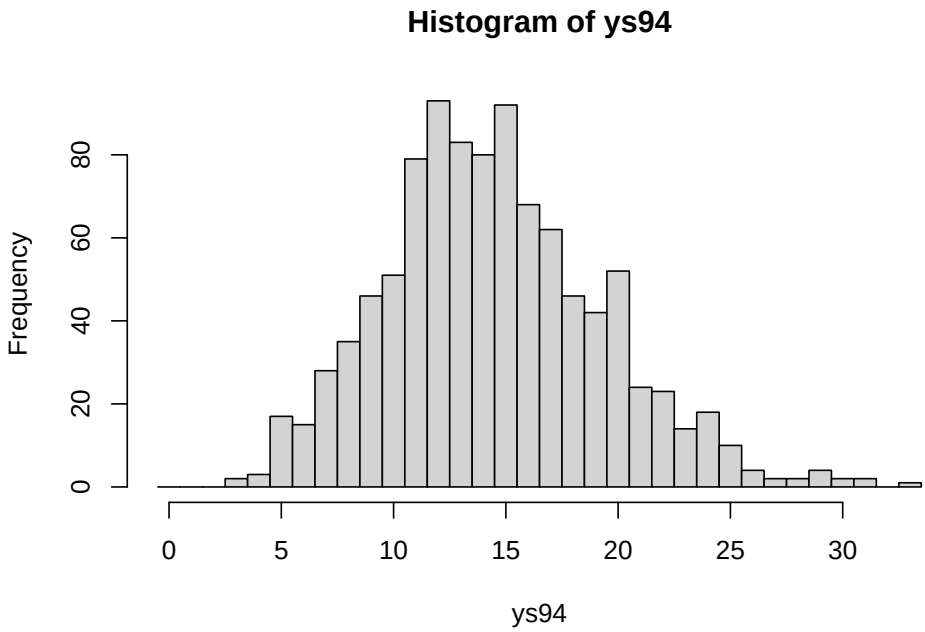
```
start <- c(4, -7)
fitgibbs <- gibbs(poissgamexch,
```

```
                  start, 1000, c(1,.15),
                  datapar)
```

```
lam94 <- with(hearttransplants,
      rgamma(1000, y[94] + alpha,
             e[94] + alpha / mu))
```

```
ys94 <- with(hearttransplants,
           rpois(1000, e[94] * lam94))
```

```
hist(ys94, breaks=seq(-0.5, max(ys94) + 0.5))
lines(y[94] * c(1, 1), c(0, 100), lwd=3)
```

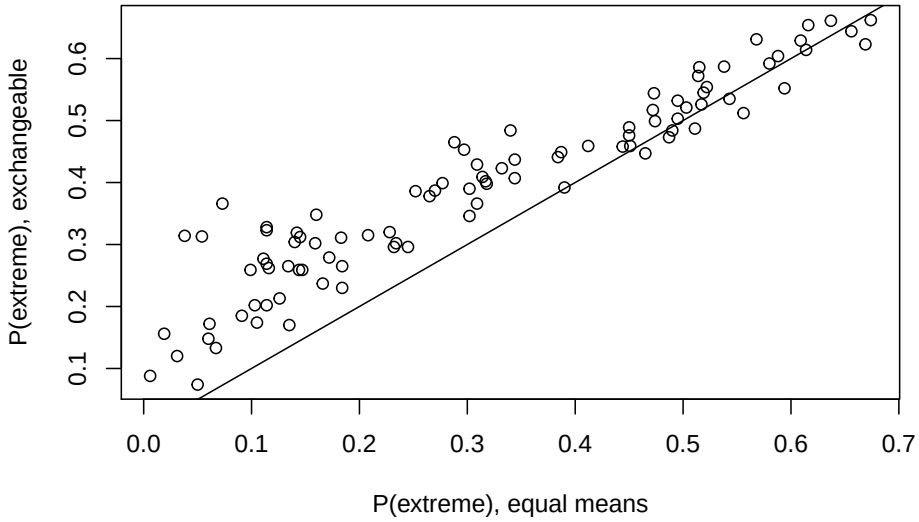


Explore the probabilities that the predictive distribution of each observation is at least as large as observed y_i.

```
prob.out <- function(i){
   lami <- with(hearttransplants,
      rgamma(1000, y[i] + alpha,
             e[i] + alpha / mu))
   ysi <- with(hearttransplants,
      rpois(1000, e[i] * lami))
   pleft <- with(hearttransplants,
      sum(ysi <= y[i]) / 1000)
   pright <- with(hearttransplants,
      sum(ysi >= y[i]) / 1000)
```

```
   min(pleft, pright)
 }
pout.exchange <- sapply(1:94, prob.out)
```

```
plot(pout, pout.exchange,
     xlab="P(extreme), equal means",
     ylab="P(extreme), exchangeable")
abline(0,1)
```



Chapter 8

Model Comparison

8.1 A One-Sided Test of a Normal Mean

Bayesian testing of $\mu \leq \mu_0$ against $\mu > \mu_0$.

```
library(LearnBayes)
```

```
pmean <- 170
pvar <- 25
probH <- pnorm(175, pmean, sqrt(pvar))
probA <- 1 - probH
prior.odds <- probH / probA
prior.odds
```

```
## [1] 5.302974
```

```
weights <- c(182, 172, 173, 176, 176, 180,
             173, 174, 179, 175)
xbar <- mean(weights)
sigma2 <- 3 ^ 2 / length(weights)
```

```
post.precision <- 1 / sigma2 + 1 / pvar
post.var <- 1 / post.precision
```

```
post.mean <- (xbar / sigma2 + pmean / pvar) /
          post.precision
c(post.mean, sqrt(post.var))
```

```
## [1] 175.7915058   0.9320546
```

```
post.odds <- pnorm(175, post.mean,
                   sqrt(post.var)) /
           (1 - pnorm(175, post.mean,
```

```
                    sqrt(post.var)))
post.odds
```

```
## [1] 0.2467017
```

```
BF <- post.odds / prior.odds
BF
```

```
## [1] 0.04652139
```

```
postH  <- probH * BF / (probH * BF + probA)
postH
```

```
## [1] 0.1978835
```

Contrast with a frequentist p-value calculation.

```
z <- sqrt(length(weights)) *
  (mean(weights) - 175) / 3
1 - pnorm(z)
```

```
## [1] 0.1459203
```

```
weights <- c(182, 172, 173, 176, 176, 180,
          173, 174, 179, 175)
data <- c(mean(weights), length(weights), 3)
prior.par <- c(170, 1000)
mnormt.onesided(175, prior.par, data)
```

```
## $BF
## [1] 0.1694947
##
## $prior.odds
## [1] 1.008011
##
## $post.odds
## [1] 0.1708525
##
## $postH
## [1] 0.1459215
```

8.2 A Two-Sided Test of a Normal Mean

Bayesian testing of $\mu = \mu_0$ against $\mu \neq \mu_0$.

```
weights <- c(182, 172, 173, 176, 176, 180,
             173, 174, 179, 175)
```

```
data <- c(mean(weights), length(weights), 3)
t <- c(.5, 1, 2, 4, 8)
mnormt.twosided(170, .5, t, data)
```

```
## $bf
## [1] 1.462146e-02 3.897038e-05 1.894326e-07 2.591162e-08 2.309739e-08
##
## $post
## [1] 1.441076e-02 3.896887e-05 1.894325e-07 2.591162e-08 2.309739e-08
```

8.3 Models for Soccer Goals

Illustrates the use of the marginal likelihood to compare several Bayesian models for soccer goals.

```
datapar <- list(data=soccergoals$goals,
                par=c(4.57, 1.43))
fit1 <- laplace(logpoissgamma, .5, datapar)
datapar <- list(data=soccergoals$goals,
                par=c(1, .5))
fit2 <- laplace(logpoissnormal, .5, datapar)
datapar <- list(data=soccergoals$goals,
                par=c(2, .5))
fit3 <- laplace(logpoissnormal, .5, datapar)
datapar <- list(data=soccergoals$goals,
                par=c(1, 2))
fit4 <- laplace(logpoissnormal, .5, datapar)
```

```
postmode <- c(fit1$mode, fit2$mode, fit3$mode,
              fit4$mode)
postsd <- sqrt(c(fit1$var, fit2$var, fit3$var,
                 fit4$var))
logmarg <- c(fit1$int, fit2$int, fit3$int,
             fit4$int)
cbind(postmode,postsd,logmarg)
```

```
##       postmode    postsd   logmarg
## [1,] 0.5248047 0.1274414 -1.502977
## [2,] 0.5207825 0.1260712 -1.255171
## [3,] 0.5825195 0.1224723 -5.076316
## [4,] 0.4899414 0.1320165 -2.137216
```

8.4 Is a Baseball Hitter Really Streaky?

Defines a family of streaky models to measure the level of support for streakiness by a Bayes factor.

```
data <- cbind(jeter2004$H, jeter2004$AB)
data1 <- regroup(data, 5)
```

```
log.marg <- function(logK){
     laplace(bfexch, 0,
          list(data=data1, K=exp(logK)))$int
}
```

```
log.K <- seq(2, 6)
K <- exp(log.K)
log.BF <- sapply(log.K, log.marg)
BF <- exp(log.BF)
round(data.frame(log.K, K, log.BF, BF), 2)
```

```
##   log.K      K log.BF   BF
## 1     2   7.39  -4.04 0.02
## 2     3  20.09   0.17 1.19
## 3     4  54.60   0.92 2.51
## 4     5 148.41   0.57 1.78
## 5     6 403.43   0.26 1.29
```

8.5 A Test of Independence in a Two-Way Contingency Table

Constructs several Bayes factor statistics for two-way contingency tables.

```
data <- matrix(c(11, 9, 68, 23, 3, 5),
               c(2, 3))
data
```

```
##      [,1] [,2] [,3]
## [1,]   11   68    3
## [2,]    9   23    5
```

Traditional chi-square test of independence.

```
chisq.test(data)
```

```
## Warning in chisq.test(data): Chi-squared approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  data
```

```
## X-squared = 6.9264, df = 2, p-value = 0.03133
```

Bayes factor against independence using uniform priors.

```
a=matrix(rep(1, 6), c(2, 3))
a
```

```
##      [,1] [,2] [,3]
## [1,]    1    1    1
## [2,]    1    1    1
```

```
ctable(data, a)
```

```
## [1] 1.662173
```

Consider Bayes factors against independence for alternatives close to independence.

```
log.K <- seq(2,7)
compute.log.BF <- function(log.K){
     log(bfindep(data, exp(log.K), 100000)$bf)
}
log.BF <- sapply(log.K, compute.log.BF)
BF <- exp(log.BF)
```

```
round(data.frame(log.K, log.BF, BF), 2)
```

```
##   log.K log.BF   BF
## 1     2  -0.59 0.55
## 2     3   0.26 1.30
## 3     4   0.77 2.16
## 4     5   0.73 2.08
## 5     6   0.43 1.54
## 6     7   0.20 1.22
```

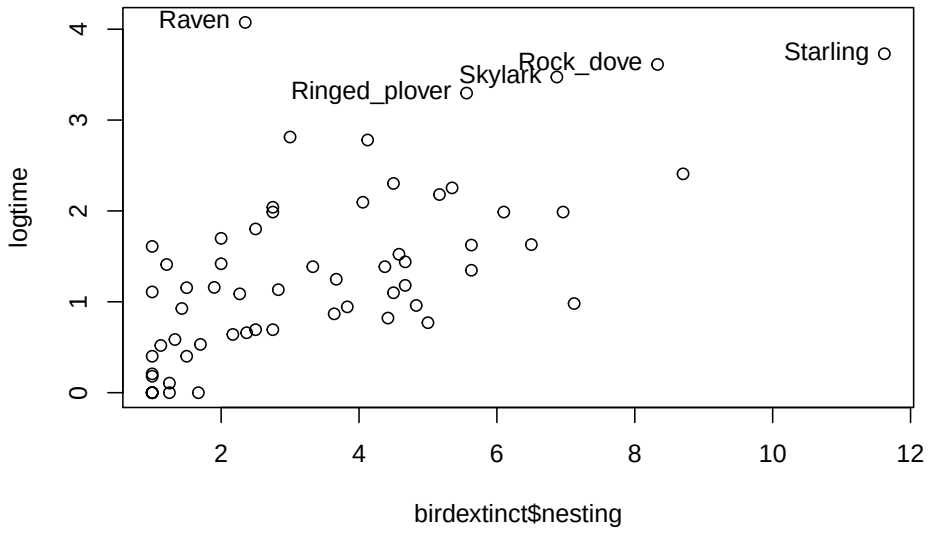
Chapter 9

Regression Models

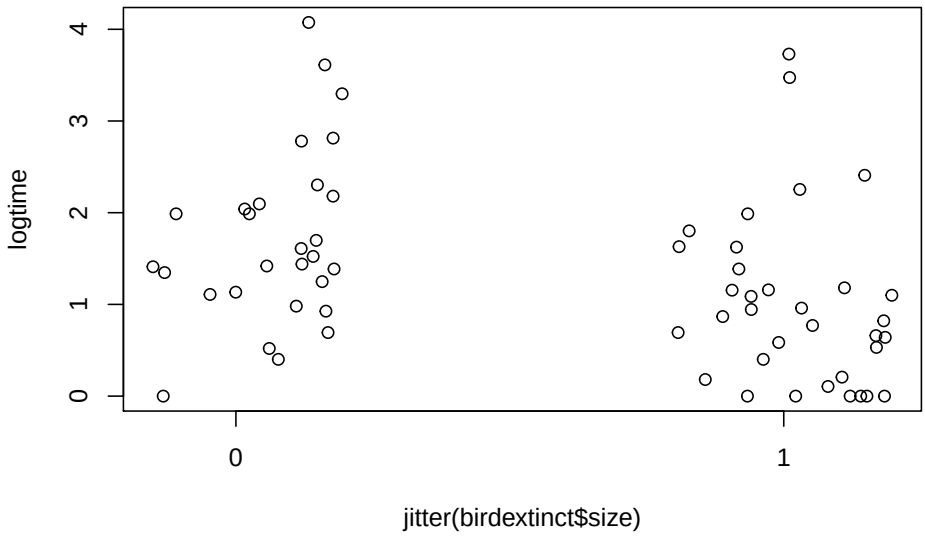
9.1 An Example of Bayesian Regression

```
library(LearnBayes)
```

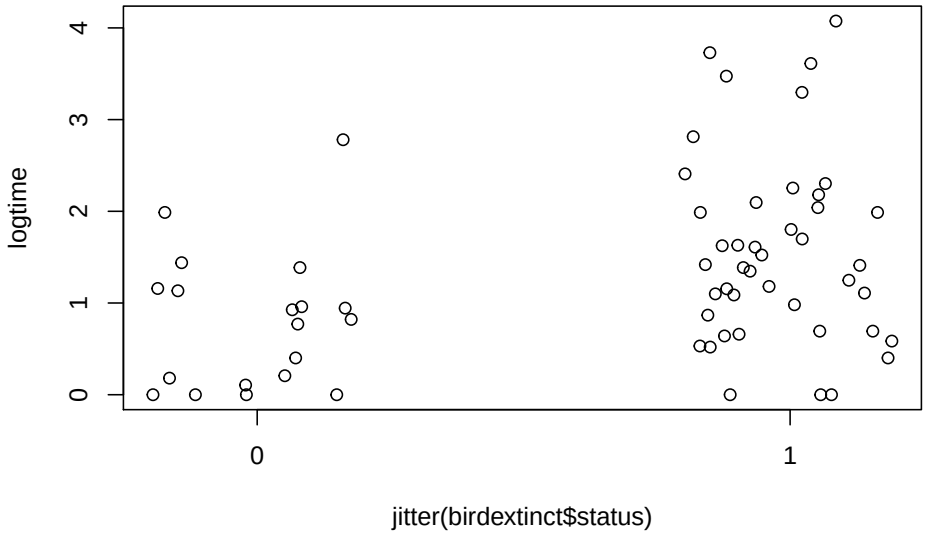
```
logtime <- log(birdextinct$time)
plot(birdextinct$nesting, logtime)
out <- (logtime > 3)
text(birdextinct$nesting[out], logtime[out],
     label=birdextinct$species[out], pos = 2)
```




```
plot(jitter(birdextinct$size), logtime,
     xaxp=c(0, 1, 1))
```




```
plot(jitter(birdextinct$status), logtime,
     xaxp=c(0, 1, 1))
```



Least-squares fit:

```
fit <- lm(logtime ~ nesting + size + status,
          data=birdextinct, x=TRUE, y=TRUE)
summary(fit)
```

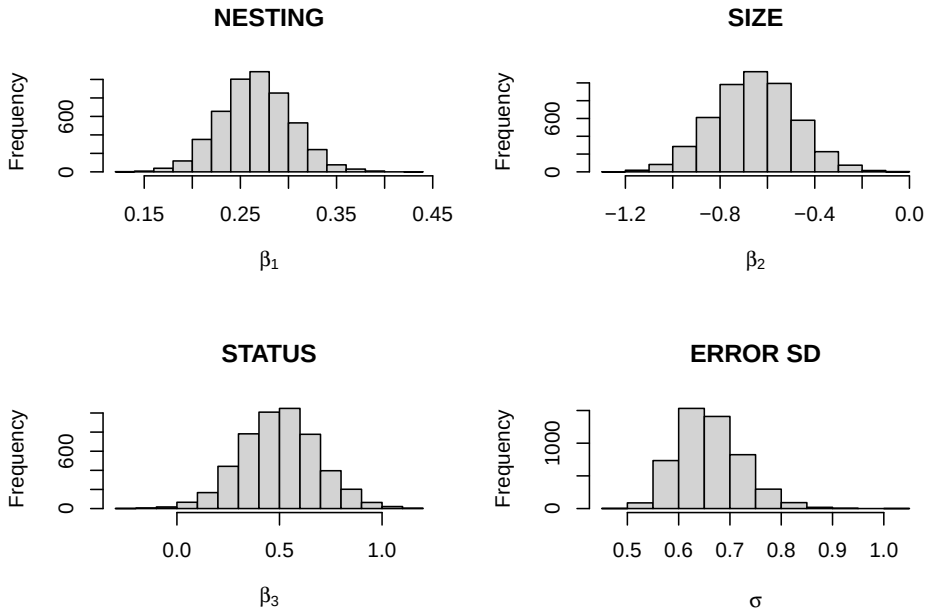
```
##
## Call:
## lm(formula = logtime ~ nesting + size + status, data = birdextinct,
```

```
##     x = TRUE, y = TRUE)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8410 -0.2932 -0.0709  0.2165  2.5167
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.43087    0.20706   2.081 0.041870 *
## nesting      0.26501    0.03679   7.203 1.33e-09 ***
## size        -0.65220    0.16667  -3.913 0.000242 ***
## status       0.50417    0.18263   2.761 0.007712 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6524 on 58 degrees of freedom
## Multiple R-squared:  0.5982, Adjusted R-squared:  0.5775
## F-statistic: 28.79 on 3 and 58 DF,  p-value: 1.577e-11
```

Sampling from posterior using vague priors for parameters.

```
theta.sample <- blinreg(fit$y, fit$x, 5000)
```

```
par(mfrow=c(2,2))
hist(theta.sample$beta[,2], main="NESTING",
  xlab=expression(beta[1]))
hist(theta.sample$beta[,3], main="SIZE",
  xlab=expression(beta[2]))
hist(theta.sample$beta[,4], main="STATUS",
  xlab=expression(beta[3]))
hist(theta.sample$sigma, main="ERROR SD",
  xlab=expression(sigma))
```




```
apply(theta.sample$beta, 2, quantile,
      c(.05, .5, .95))
```

```
##     X(Intercept)  Xnesting      Xsize   Xstatus
## 5%    0.09782926 0.2060394 -0.9349520 0.1973827
## 50%   0.42879649 0.2654044 -0.6539278 0.5011496
## 95%   0.78922321 0.3268835 -0.3771980 0.8145627
```

```
quantile(theta.sample$sigma, c(.05, .5, .95))
```

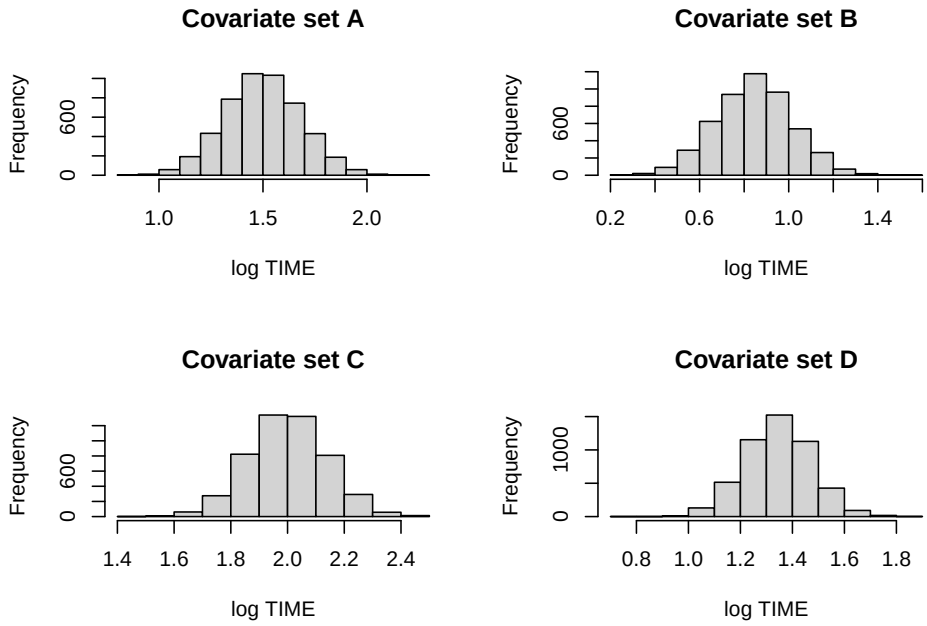
```
##        5%       50%       95%
## 0.5691949 0.6548724 0.7692441
```

Estimating mean extinction times:

```
cov1 <- c(1, 4, 0, 0)
cov2 <- c(1, 4, 1, 0)
cov3 <- c(1, 4, 0, 1)
cov4 <- c(1, 4, 1, 1)
X1 <- rbind(cov1, cov2, cov3, cov4)
mean.draws <- blinregexpected(X1, theta.sample)
```

```
c.labels <- c("A", "B", "C", "D")
par(mfrow=c(2, 2))
 for (j in 1:4){
   hist(mean.draws[, j],
      main=paste("Covariate set",
                 c.labels[j]),
```

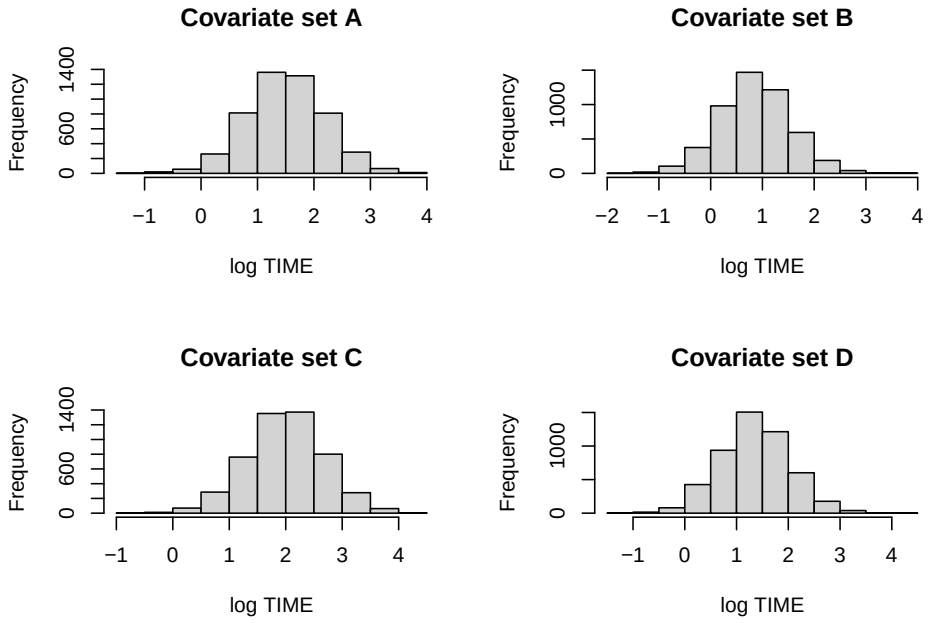
```
      xlab="log TIME")
}
```



Predicting future extinction times:

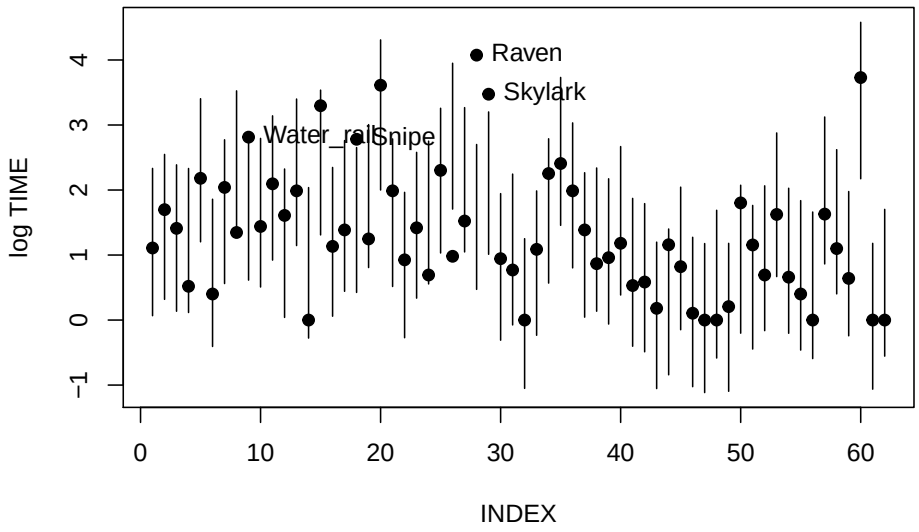
```
cov1 <- c(1, 4, 0, 0)
cov2 <- c(1, 4, 1, 0)
cov3 <- c(1, 4, 0, 1)
cov4 <- c(1, 4, 1, 1)
X1 <- rbind(cov1, cov2, cov3, cov4)
pred.draws <- blinregpred(X1, theta.sample)
```

```
c.labels <- c("A", "B", "C", "D")
par(mfrow=c(2,2))
for (j in 1:4){
   hist(pred.draws[, j],
      main=paste("Covariate set",
                 c.labels[j]),
      xlab="log TIME")
}
```



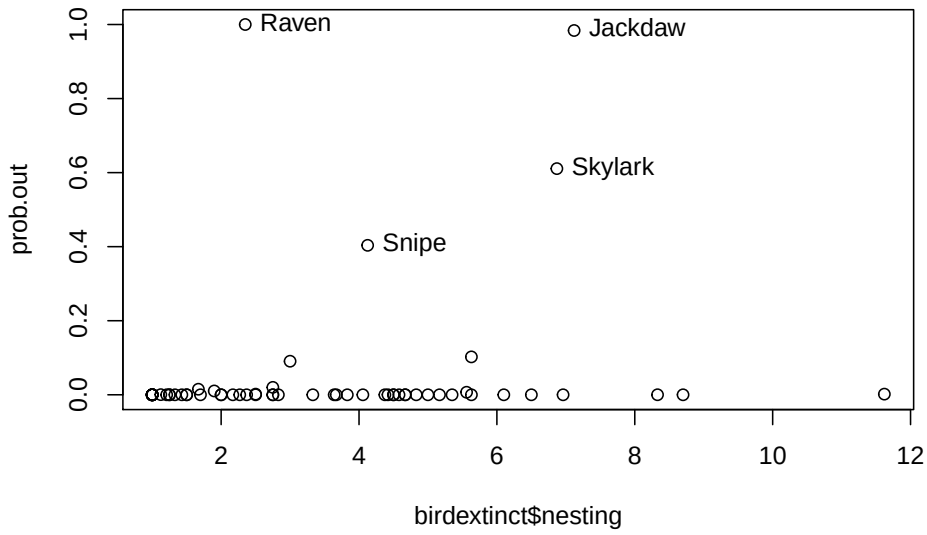
Model checking: posterior predictive distribution distributions of each future observation, showing actual observation as solid dot.

```
pred.draws <- blinregpred(fit$x, theta.sample)
pred.sum <- apply(pred.draws, 2,
                  quantile, c(.05,.95))
par(mfrow=c(1, 1))
ind <- 1:length(logtime)
matplot(rbind(ind, ind), pred.sum,
        type="l", lty=1, col=1,
        xlab="INDEX", ylab="log TIME")
points(ind, logtime, pch=19)
out <- (logtime > pred.sum[2, ])
text(ind[out], logtime[out],
     label=birdextinct$species[out], pos = 4)
```



Model checking via bayes residuals $y_i - x_i\beta$. Graph of absolute values of residuals that exceeds a particular constant.

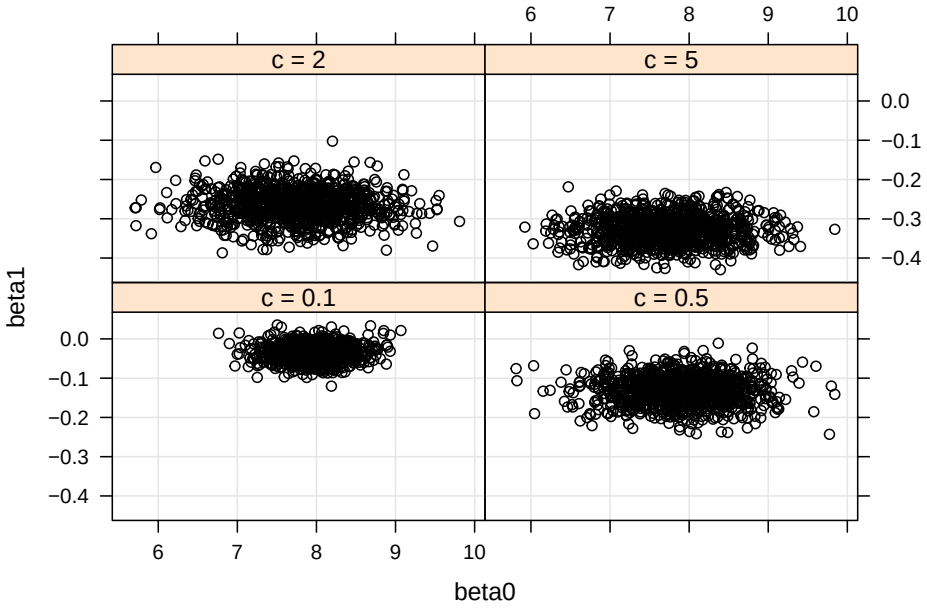
```
prob.out <- bayesresiduals(fit, theta.sample, 2)
par(mfrow=c(1, 1))
plot(birdextinct$nesting, prob.out)
out = (prob.out > 0.35)
text(birdextinct$nesting[out], prob.out[out],
     label=birdextinct$species[out], pos = 4)
```



9.2 Modeling Using Zellner's g Prior

Illustrating the role of the parameter c:

```
X <- cbind(1, puffin$Distance -
             mean(puffin$Distance))
c.prior <- c(0.1, 0.5, 5, 2)
fit <- vector("list", 4)
for (j in 1:4){
  prior <- list(b0=c(8, 0), c0=c.prior[j])
  fit[[j]] <- blinreg(puffin$Nest, X, 1000, prior)
}
BETA <- NULL
for (j in 1:4){
  s=data.frame(Prior=paste("c =",
                 as.character(c.prior[j])),
         beta0=fit[[j]]$beta[, 1],
         beta1=fit[[j]]$beta[, 2])
  BETA <- rbind(BETA, s)
}
library(lattice)
with(BETA,
     xyplot(beta1 ~ beta0 | Prior,
            type=c("p","g"),
            col="black"))
```



Model selection of all regression models using g priors:

```
data <- list(y=puffin$Nest,
             X=cbind(1, puffin$Grass, puffin$Soil))
prior <- list(b0=c(0, 0, 0), c0=100)
beta.start <- with(puffin,
          lm(Nest ~ Grass + Soil)$coef)
laplace(reg.gprior.post,
        c(beta.start, 0),
        list(data=data, prior=prior))$int
```

```
## [1] -136.3957
```

```
X <- puffin[, -1]
y <- puffin$Nest
c <- 100
bayes.model.selection(y, X, c, constant=FALSE)
```

```
## $mod.prob
##    Grass  Soil Angle Distance   log.m    Prob
## 1  FALSE FALSE FALSE    FALSE -132.18 0.00000
## 2   TRUE FALSE FALSE    FALSE -134.05 0.00000
## 3  FALSE  TRUE FALSE    FALSE -134.51 0.00000
## 4   TRUE  TRUE FALSE    FALSE -136.40 0.00000
## 5  FALSE FALSE  TRUE    FALSE -112.67 0.00000
## 6   TRUE FALSE  TRUE    FALSE -113.18 0.00000
## 7  FALSE  TRUE  TRUE    FALSE -114.96 0.00000
## 8   TRUE  TRUE  TRUE    FALSE -115.40 0.00000
## 9  FALSE FALSE FALSE     TRUE -103.30 0.03500
## 10  TRUE FALSE FALSE     TRUE -105.57 0.00360
## 11 FALSE  TRUE FALSE     TRUE -100.37 0.65065
## 12  TRUE  TRUE FALSE     TRUE -102.35 0.08992
## 13 FALSE FALSE  TRUE     TRUE -102.81 0.05682
## 14  TRUE FALSE  TRUE     TRUE -105.09 0.00581
## 15 FALSE  TRUE  TRUE     TRUE -101.88 0.14386
## 16  TRUE  TRUE  TRUE     TRUE -104.19 0.01434
##
## $converge
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [16] TRUE
```

9.3 Survival Modeling

Traditional fit using a Weibull model:

```
library(survival)
survreg(Surv(time, status) ~ factor(treat) + age,
        dist="weibull",
```

```
        data = chemotherapy)
```

```
## Call:
## survreg(formula = Surv(time, status) ~ factor(treat) + age, data = chemotherapy,
##     dist = "weibull")
##
## Coefficients:
##    (Intercept) factor(treat)2            age
##    10.98683919     0.56145663    -0.07897718
##
## Scale= 0.5489202
##
## Loglik(model)= -88.7   Loglik(intercept only)= -98
##  Chisq= 18.41 on 2 degrees of freedom, p= 0.000101
## n= 26
```

Bayesian fit:

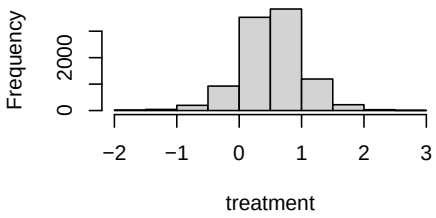
```
start <- c(-.5, 9, .5, -.05)
d <- with(chemotherapy,
          cbind(time, status, treat - 1, age))
fit <- laplace(weibullregpost, start, d)
fit
```

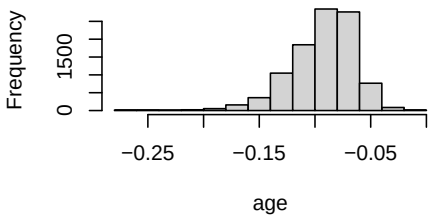
```
## $mode
## [1] -0.59986796 10.98663371  0.56151088 -0.07897316
##
## $var
##               [,1]        [,2]         [,3]          [,4]
## [1,]  0.057298875  0.13530436  0.004541435 -0.0020828431
## [2,]  0.135304360  1.67428176 -0.156631948 -0.0255278352
## [3,]  0.004541435 -0.15663195  0.115450201  0.0017880712
## [4,] -0.002082843 -0.02552784  0.001788071  0.0003995202
##
## $int
## [1] -25.31207
##
## $converge
## [1] TRUE
```

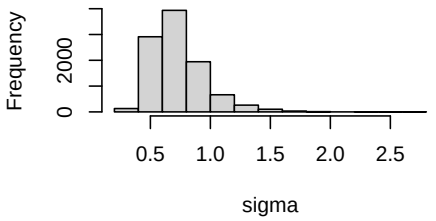
```
proposal <- list(var=fit$var, scale=1.5)
bayesfit <- rwmetrop(weibullregpost,
                     proposal,
                     fit$mode,
                     10000, d)
bayesfit$accept
```

```
## [1] 0.271
```

```
par(mfrow=c(2, 2))
sigma <- exp(bayesfit$par[, 1])
mu <- bayesfit$par[, 2]
beta1 <- bayesfit$par[, 3]
beta2 <- bayesfit$par[, 4]
hist(beta1, xlab="treatment", main="")
hist(beta2, xlab="age", main="")
hist(sigma, xlab="sigma", main="")
```







Chapter 10

Gibbs Sampling

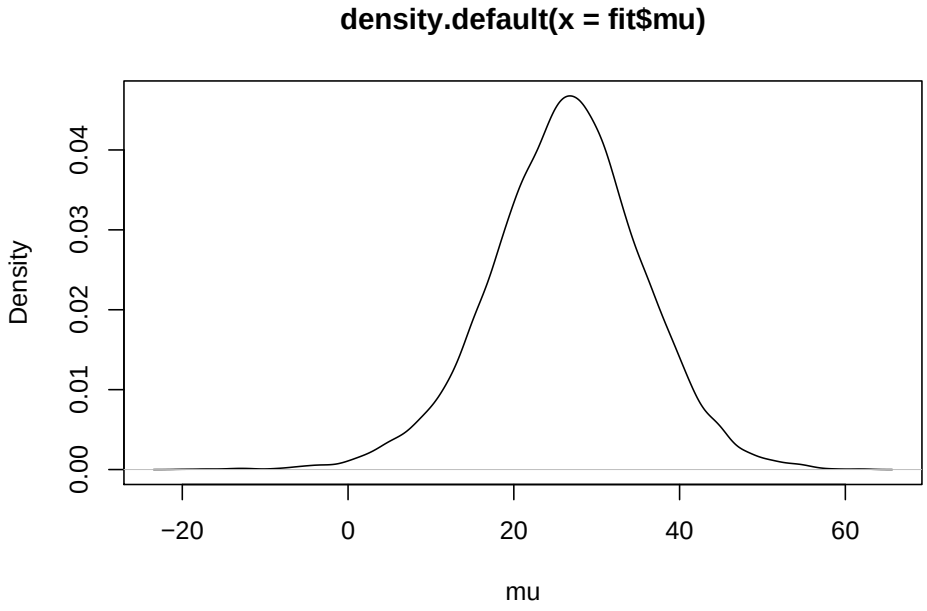
10.1 Robust Modeling

Illustrating Gibbs sampling using a t sampling model.

```
library(LearnBayes)
```

```
fit <- robustt(darwin$difference, 4, 10000)
```

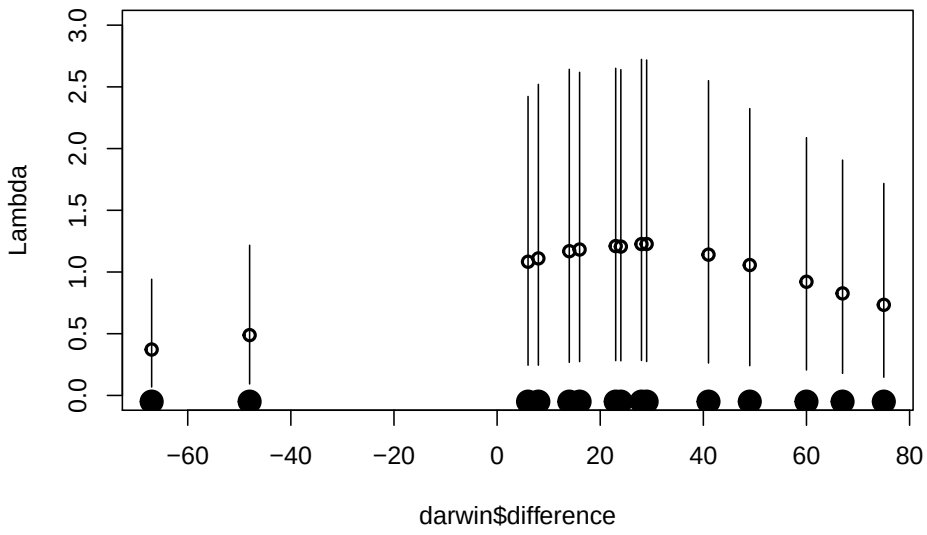
```
plot(density(fit$mu), xlab="mu")
```



The λ_j parameters indicate the outlying observations.

```
mean.lambda <- apply(fit$lam, 2, mean)
lam5 <- apply(fit$lam, 2, quantile, .05)
lam95 <- apply(fit$lam, 2, quantile, .95)
```

```
plot(darwin$difference, mean.lambda,
     lwd=2, ylim=c(0,3), ylab="Lambda")
for (i in 1:length(darwin$difference)){
   lines(c(1, 1) * darwin$difference[i],
         c(lam5[i], lam95[i]))
}
points(darwin$difference,
       0 * darwin$difference-.05,
       pch=19, cex=2)
```



10.2 Binary Response Regression with a Probit Link

Missing data and Gibbs sampling

```
X <- with(donner,
          cbind(1, age, male))
```

Traditional probit fit:

```
fit <- glm(survival ~ X - 1,
           family=binomial(link=probit),
           data = donner)
summary(fit)
```

```
##
## Call:
## glm(formula = survival ~ X - 1, family = binomial(link = probit),
##     data = donner)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7420  -1.0555  -0.2756   0.8861   2.0339
##
## Coefficients:
##       Estimate Std. Error z value Pr(>|z|)
## X      1.91730    0.76438   2.508   0.0121 *
## Xage  -0.04571    0.02076  -2.202   0.0277 *
## Xmale -0.95828    0.43983  -2.179   0.0293 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 62.383  on 45  degrees of freedom
## Residual deviance: 51.283  on 42  degrees of freedom
## AIC: 57.283
##
## Number of Fisher Scoring iterations: 5
```

Bayesian fit of the probit model using data augmentation.

```
m <- 10000
fit <- bayes.probit(donner$survival, X, m)
```

```
apply(fit$beta,2,mean)
```

```
## [1]  2.08778647 -0.05013532 -1.02079631
```

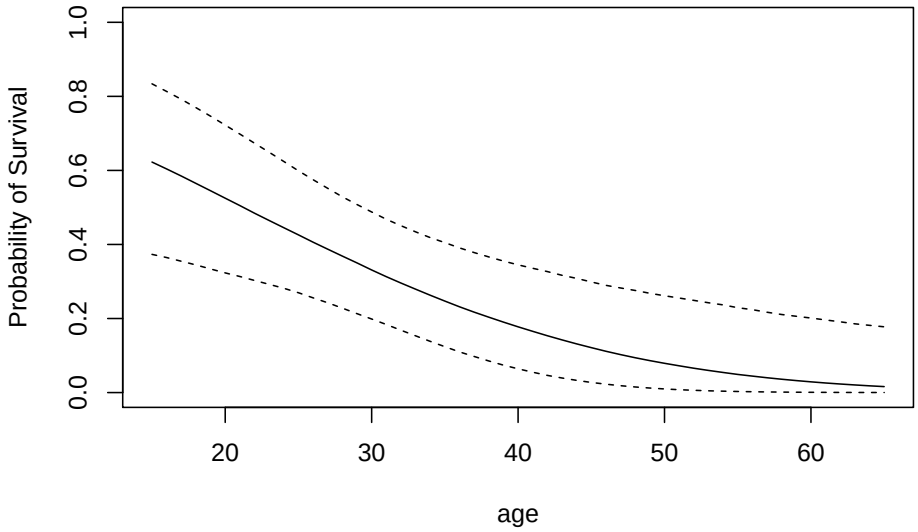
```
apply(fit$beta,2,sd)
```

```
## [1] 0.81408634 0.02171058 0.45311813
```

Posterior distributions of specific probabilities.

```
a <- seq(15, 65)
X1 <- cbind(1, a, 1)
p.male <- bprobit.probs(X1, fit$beta)
```

```
plot(a, apply(p.male, 2, quantile, .5),
     type="l", ylim=c(0,1),
   xlab="age", ylab="Probability of Survival")
lines(a,apply(p.male, 2, quantile, .05), lty=2)
lines(a,apply(p.male, 2, quantile, .95), lty=2)
```



Proper priors and model selection of probit models.

```
y <- donner$survival
X <- cbind(1, donner$age, donner$male)
```

```
beta0 <- c(0,0,0)
c0 <- 100
P0 <- t(X) %*% X / c0
```

```
bayes.probit(y, X, 1000,
             list(beta=beta0, P=P0))$log.marg
```

```
## [1] -31.5737
```

```
bayes.probit(y, X[, -2], 1000,
   list(beta=beta0[-2], P=P0[-2, -2]))$log.marg
```

```
## [1] -32.77612
```

```
bayes.probit(y, X[, -3], 1000,
   list(beta=beta0[-3], P=P0[-3, -3]))$log.marg
```

```
## [1] -32.06246
```

```
bayes.probit(y, X[, -c(2, 3)], 1000,
              list(beta=beta0[- c(2, 3)],
      P=P0[-c(2, 3), -c(2, 3)]))$log.marg
```

```
## [1] -32.98507
```

10.3 Estimating a Table of Means

```
rlabels <- c("91-99", "81-90", "71-80",
             "61-70", "51-60", "41-50",
             "31-40", "21-30")
clabels <- c("16-18", "19-21", "22-24",
             "25-27", "28-30")
gpa <- matrix(iowagpa[, 1],
              nrow = 8, ncol = 5, byrow = T)
dimnames(gpa) <- list(HSR = rlabels,
                      ACTC = clabels)
gpa
```

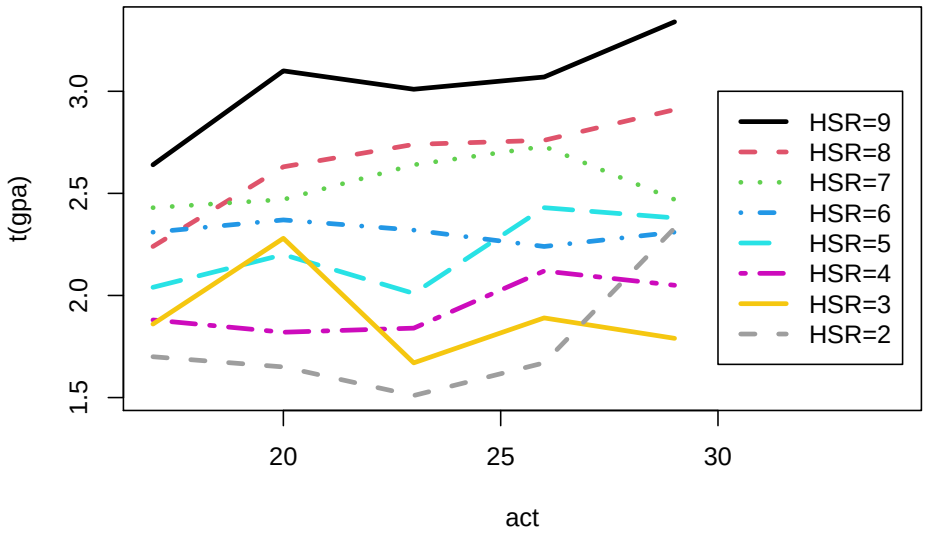
```
##        ACTC
## HSR     16-18 19-21 22-24 25-27 28-30
##   91-99  2.64  3.10  3.01  3.07  3.34
##   81-90  2.24  2.63  2.74  2.76  2.91
##   71-80  2.43  2.47  2.64  2.73  2.47
##   61-70  2.31  2.37  2.32  2.24  2.31
##   51-60  2.04  2.20  2.01  2.43  2.38
##   41-50  1.88  1.82  1.84  2.12  2.05
##   31-40  1.86  2.28  1.67  1.89  1.79
##   21-30  1.70  1.65  1.51  1.67  2.33
```

```
samplesizes <- matrix(iowagpa[, 2],
                  nrow = 8, ncol = 5, byrow = T)
dimnames(samplesizes) <- list(HSR = rlabels,
                              ACTC = clabels)
samplesizes
```

```
##        ACTC
## HSR     16-18 19-21 22-24 25-27 28-30
##   91-99     8    15    78   182   166
##   81-90    20    71   168   178    91
##   71-80    40   116   180   133    46
##   61-70    34    93   124   101    19
##   51-60    41    73    62    58     9
##   41-50    19    25    36    49    16
##   31-40     8     9    15    29     9
##   21-30     4     5     9    11     1
```

```
act <- seq(17, 29, by = 3)
matplot(act, t(gpa), type = "l", lwd = 3,
  xlim = c(17, 34), col=1:8, lty=1:8)
legend(30, 3, lty = 1:8, lwd = 3,
       legend = c("HSR=9", "HSR=8",
     "HSR=7", "HSR=6", "HSR=5", "HSR=4",
```

```
      "HSR=3", "HSR=2"), col=1:8)
```



Fitting a Bayesian model with a flat prior over the restricted space.

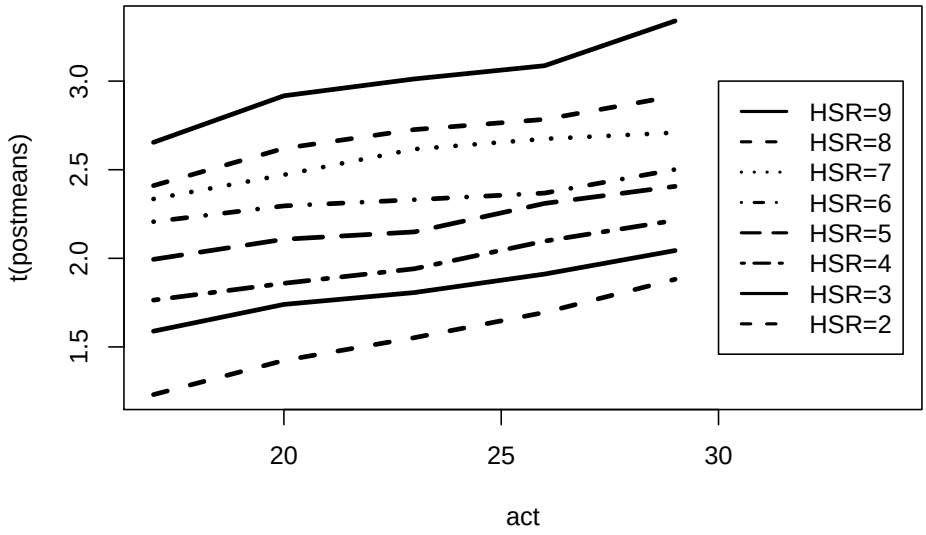
```
MU <- ordergibbs(iowagpa, 5000)
```

```
postmeans <- apply(MU, 2, mean)
postmeans <- matrix(postmeans, nrow = 8, ncol = 5)
postmeans <- postmeans[seq(8, 1, -1), ]
dimnames(postmeans) <-
  list(HSR=rlabels, ACTC=clabels)
round(postmeans, 2)
```

```
##        ACTC
## HSR     16-18 19-21 22-24 25-27 28-30
##   91-99  2.65  2.92  3.01  3.09  3.34
##   81-90  2.41  2.62  2.73  2.78  2.92
##   71-80  2.33  2.47  2.62  2.67  2.71
##   61-70  2.21  2.30  2.33  2.37  2.50
##   51-60  1.99  2.11  2.15  2.31  2.41
##   41-50  1.76  1.86  1.94  2.10  2.21
##   31-40  1.59  1.74  1.81  1.91  2.04
##   21-30  1.23  1.42  1.55  1.69  1.88
```

```
matplot(act, t(postmeans), type = "l",
        lty=1:8, lwd = 3, col = 1,
        xlim = c(17, 34))
legend(30, 3, lty = 1:8, lwd = 2,
       legend = c("HSR=9", "HSR=8",
     "HSR=7", "HSR=6", "HSR=5", "HSR=4",
```

```
     "HSR=3", "HSR=2"))
```




```
postsds <- apply(MU, 2, sd)
postsds <- matrix(postsds, nrow = 8, ncol = 5)
postsds <- postsds[seq(8, 1, -1), ]
dimnames(postsds) <- list(HSR=rlabels,
                          ACTC=clabels)
round(postsds, 3)
```

```
##         ACTC
## HSR      16-18 19-21 22-24 25-27 28-30
##   91-99 0.141 0.085 0.054 0.043 0.051
##   81-90 0.075 0.059 0.038 0.038 0.063
##   71-80 0.064 0.051 0.038 0.039 0.047
##   61-70 0.066 0.039 0.036 0.038 0.080
##   51-60 0.076 0.053 0.055 0.049 0.075
##   41-50 0.082 0.067 0.067 0.071 0.086
##   31-40 0.115 0.078 0.072 0.075 0.099
##   21-30 0.183 0.139 0.118 0.113 0.131
```

```
s <- .65
se <- s / sqrt(samplesizes)
round(postsds / se, 2)
```

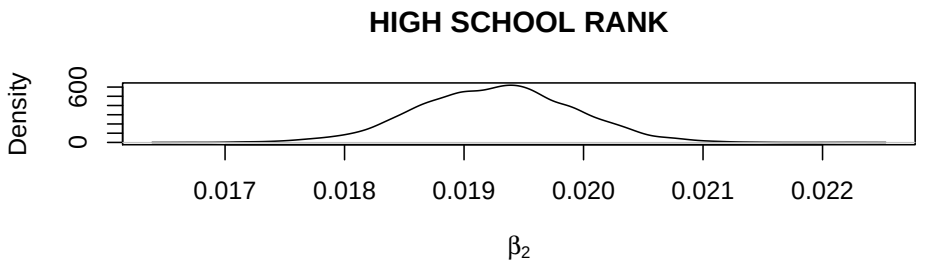
```
##         ACTC
## HSR      16-18 19-21 22-24 25-27 28-30
##   91-99  0.61  0.51  0.74  0.90  1.00
##   81-90  0.52  0.76  0.75  0.79  0.92
##   71-80  0.62  0.85  0.78  0.69  0.49
```

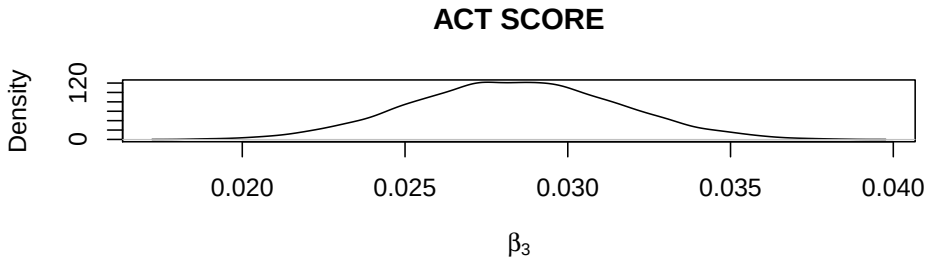
```
##   61-70  0.59  0.59  0.62  0.59  0.53
##   51-60  0.75  0.70  0.66  0.57  0.34
##   41-50  0.55  0.52  0.62  0.76  0.53
##   31-40  0.50  0.36  0.43  0.62  0.46
##   21-30  0.56  0.48  0.54  0.58  0.20
```

Fit of a hierarchical regression prior:

```
FIT <- hiergibbs(iowagpa, 5000)
```

```
par(mfrow=c(2,1))
plot(density(FIT$beta[, 2]),
     xlab=expression(beta[2]),
     main="HIGH SCHOOL RANK")
plot(density(FIT$beta[, 3]),
     xlab=expression(beta[3]),
     main="ACT SCORE")
```






```
quantile(FIT$beta[, 2],
         c(.025, .25, .5, .75, .975))
```

```
##       2.5%        25%        50%        75%      97.5%
## 0.01798691 0.01882042 0.01927662 0.01968910 0.02051474
```

```
quantile(FIT$beta[, 3],
         c(.025, .25, .5, .75, .975))
```

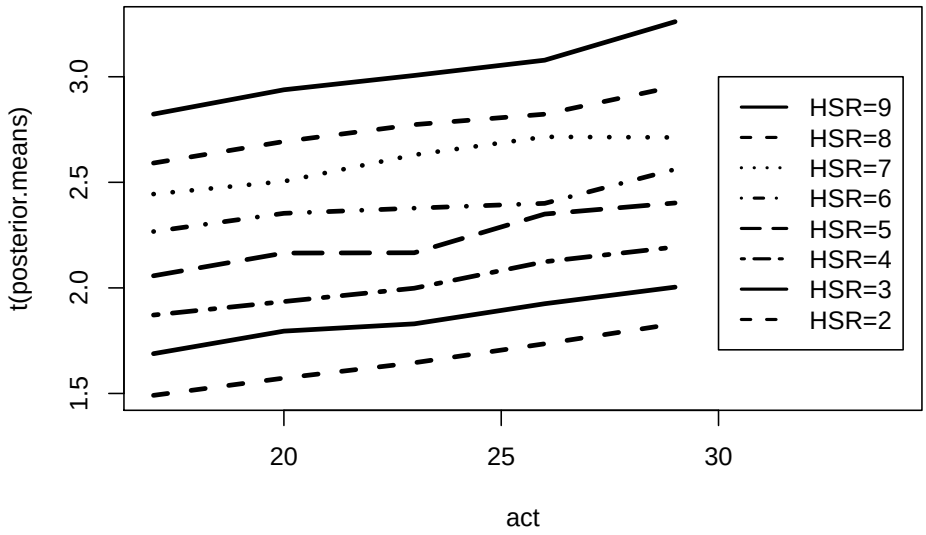
```
##       2.5%        25%        50%        75%      97.5%
## 0.02224535 0.02619811 0.02835767 0.03049379 0.03455889
```

```
quantile(FIT$var,
         c(.025, .25, .5, .75, .975))
```

```
##        2.5%         25%         50%         75%       97.5%
## 0.001099840 0.001967612 0.002783425 0.003925143 0.007472338
```

```
posterior.means <- apply(FIT$mu, 2, mean)
posterior.means <- matrix(posterior.means,
                          nrow = 8, ncol = 5,
                          byrow = T)
```

```
par(mfrow=c(1, 1))
matplot(act, t(posterior.means),
        type = "l", lwd = 3, lty=1:8, col=1,
        xlim = c(17, 34))
legend(30, 3, lty = 1:8, lwd = 2,
       legend = c("HSR=9", "HSR=8", "HSR=7",
                  "HSR=6", "HSR=5", "HSR=4",
                  "HSR=3", "HSR=2"))
```




```
p <- 1 - pnorm((2.5 - FIT$mu) / .65)
prob.success <- apply(p, 2, mean)
```

```
prob.success <- matrix(prob.success,
                       nrow=8, ncol=5, byrow=T)
dimnames(prob.success) <- list(HSR=rlabels,
                               ACTC=clabels)
round(prob.success,3)
```

```
##        ACTC
```

```
## HSR     16-18 19-21 22-24 25-27 28-30
##   91-99 0.690 0.749 0.781 0.813 0.879
##   81-90 0.556 0.617 0.663 0.690 0.757
##   71-80 0.466 0.503 0.579 0.630 0.628
##   61-70 0.360 0.411 0.425 0.439 0.538
##   51-60 0.249 0.304 0.304 0.409 0.440
##   41-50 0.168 0.193 0.221 0.282 0.319
##   31-40 0.107 0.140 0.152 0.189 0.224
##   21-30 0.061 0.078 0.095 0.121 0.153
```