```r
library(latex2exp)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
## rstan (Version 2.21.5, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
##
## Attaching package: 'rstan'
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```r
library("gridExtra")
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(doParallel)
```

```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```r
registerDoParallel()

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

```r
tumor_experiments <- read.csv("../01_data/data.csv")
```

```r
tumor_experiments$percent = tumor_experiments$tumors / tumor_experiments$n
tumor_experiments$y <- tumor_experiments$tumors
# n <- tumor_experiments$n
y <- c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,
        2,1,5,2,5,3,2,7,7,3,3,2,9,10,4,4,4,4,4,4,4,10,4,4,4,5,11,12,
        5,5,6,5,6,6,6,6,16,15,15,9,4)
n <- c(20,20,20,20,20,20,20,19,19,19,19,18,18,17,20,20,20,20,19,19,18,18,25,24,
       23,20,20,20,20,20,20,10,49,19,46,27,17,49,47,20,20,13,48,50,20,20,20,20,
       20,20,20,48,19,19,19,22,46,49,20,20,23,19,22,20,20,20,52,46,47,24,14)
j <- length(y)
```

Marginal posterior distribution (5.8) and helper functions

$$P(\alpha, \beta|\text{y}) \propto P(\alpha, \beta) \prod_j \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \frac{\Gamma(\alpha+c_j)\Gamma(\beta+n_j-c_j)}{\Gamma(\alpha+\beta+n_j)}$$

, We will compute this log-transformed.

```r
model.prior = prior = function(alpha, beta) {
    (alpha + beta) ^ (-5 / 2)
}
par.beta = function(x, y) {
    exp(y) / (exp(x) + 1)
}
par.alpha = function(x, y) {
    exp(x) * par.beta(x, y)
}

marg_post = function(alpha, beta){
  ldens <- 0
  for (i in 1:length(y)) ldens <- ldens +
    lgamma(alpha + beta) + lgamma(alpha + y[i]) + lgamma(beta + n[i] - y[i])-
                                 (lgamma(alpha) + lgamma(beta) + lgamma(alpha + beta + n[i]))
  ldens + log(alpha*beta) + log(alpha+beta)*(-5/2) }
```

figure 5.2

Here we compute the values of the log-transformed posterior density on the grid. Following the example of Gelman BA, we also subtract the maximum value from every point on the grid and exponentiate so that all values lie between 0 and 1. We then show contours at 0.1, 0.2 ... 1 times the density at the mode.

```r
xmin <- -2.5
xmax = -1
ymin<- 1.5
ymax <- 3
xcord <- seq(xmin, xmax, length.out = 100)
ycord <- seq(ymin, ymax, length.out = 100)

grid <- expand.grid(x=xcord, y=ycord) %>%
  mutate(a=par.alpha(x,y),
         b=par.beta(x,y)) %>% mutate(z=marg_post(a,b)) %>% mutate(z_rescale = exp(z - max(z)))


ggplot(grid, aes(x=x, y=y)) +
```

```
    geom_contour(aes(z=z_rescale), binwidth=0.05)  +
    xlab("log(a / b)") +
    ylab("log(a + b)") + metR::geom_text_contour(aes(z=z_rescale)) +  theme(aspect.ratio=1) + xlim(xmin
```

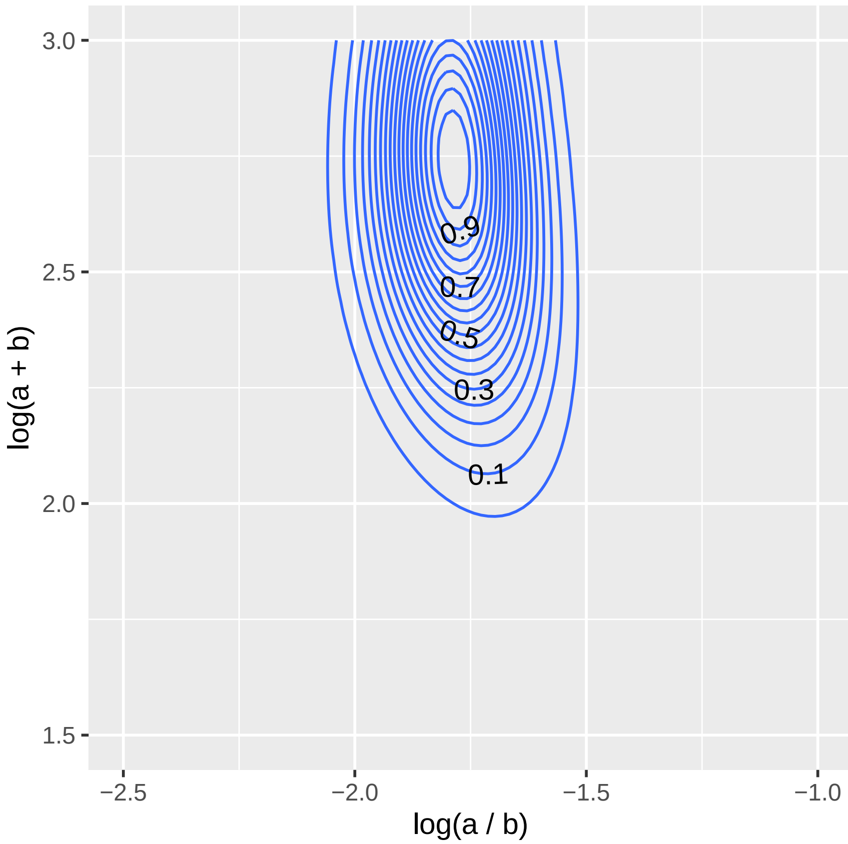


figure 5.3a

The same explanation for figure 5.2 holds but here we zoom in on the density by choosing a narrower grid.

```
xmin <- -2.3
xmax <- -1.3
ymin<- 1
ymax <- 5
xcord <- seq(xmin, xmax, length.out = 100)
ycord <- seq(ymin, ymax, length.out = 100)

grid <- expand.grid(x=xcord, y=ycord) %>%
  mutate(alpha=par.alpha(x,y),
         beta=par.beta(x,y)) %>% mutate(z=marg_post(alpha,beta)) %>% mutate(z_rescale = exp(z - max(z)))


ggplot(grid, aes(x=x, y=y)) +
    geom_contour(aes(z=z_rescale), binwidth=0.05)  +
    xlab("log(a / b)") +
    ylab("log(a + b)") + metR::geom_text_contour(aes(z=z_rescale)) +  theme(aspect.ratio=1) + xlim(xmin
```

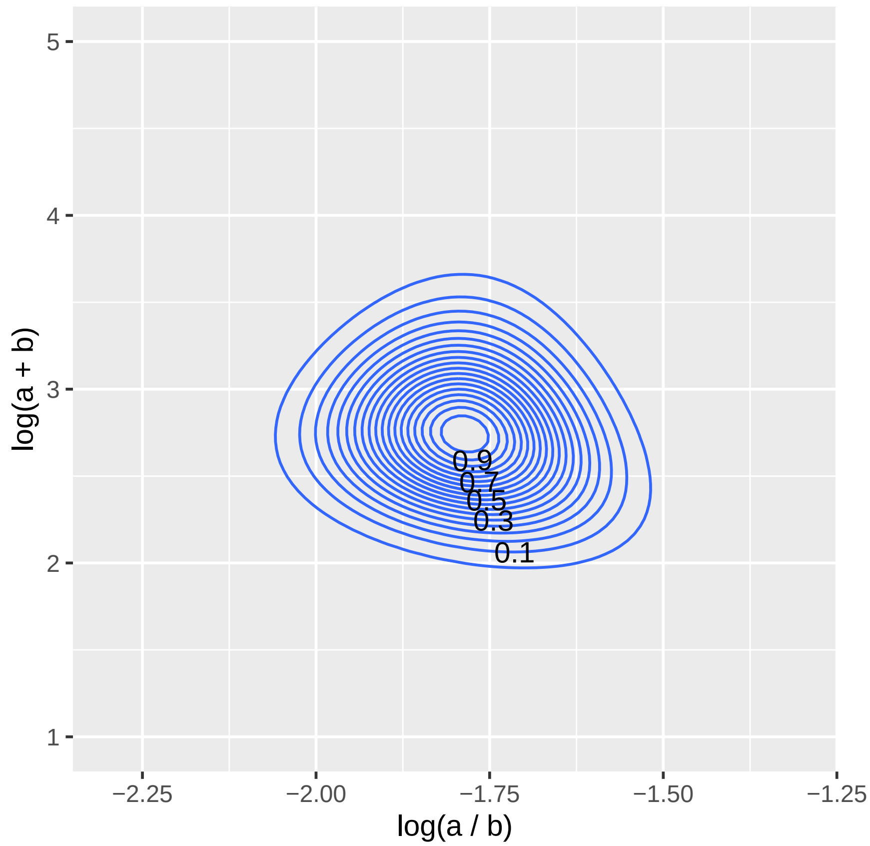


figure 5.3b

Below, we sample our log-transformed $\alpha's$ and $\beta's$ from the normalized grid. We then plot these samples in a similar fashion to figure 5.3a

```
nsamp <- 1000
p <- grid$z_rescale

samp_indices <- sample(length(p), size = nsamp,
                       replace = T, prob = p/sum(p))

samp_x <- grid$x[samp_indices[1:nsamp]]
samp_y <- grid$y[samp_indices[1:nsamp]]

samp_x <- samp_x + runif(nsamp, (xcord[1] - xcord[2])/2, (xcord[2] - xcord[1])/2)
samp_y <- samp_y + runif(nsamp, (ycord[1] - ycord[2])/2, (ycord[2] - ycord[1])/2)
samps <- tibble(ind = 1:nsamp, x = samp_x, y = samp_y)

sam <- ggplot(data = samps) +
  geom_point(aes(x, y), color = 'blue') +
  labs(title = 'Posterior draws', x = "log(a / b)", y = "log(a + b)") +
  theme(aspect.ratio=1) + xlim(xmin, xmax) + ylim(ymin, ymax)
sam
```

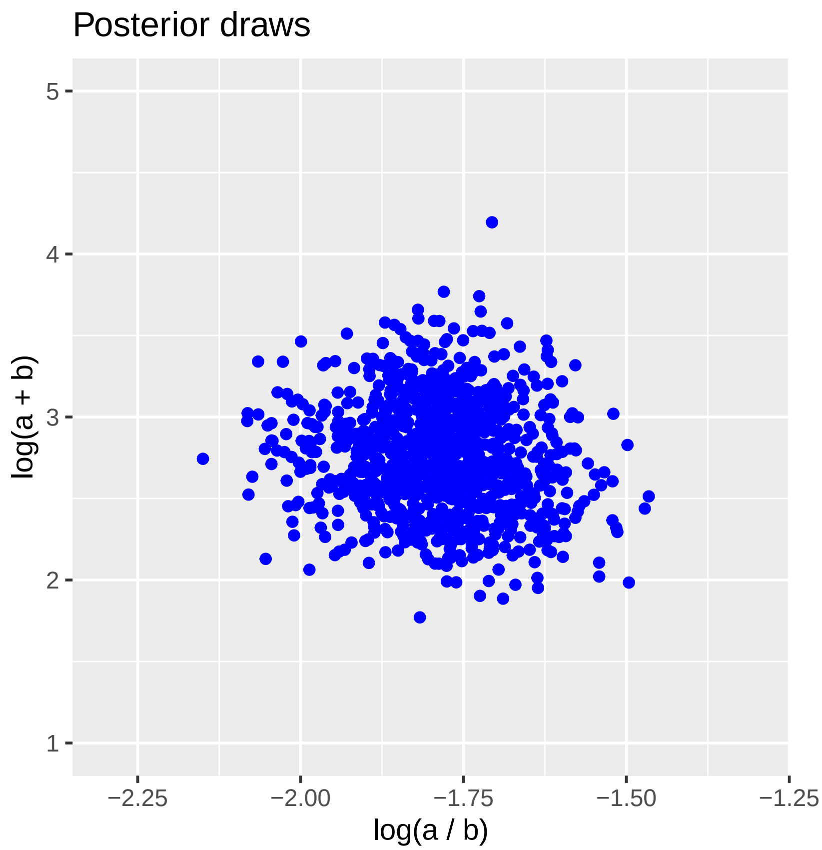
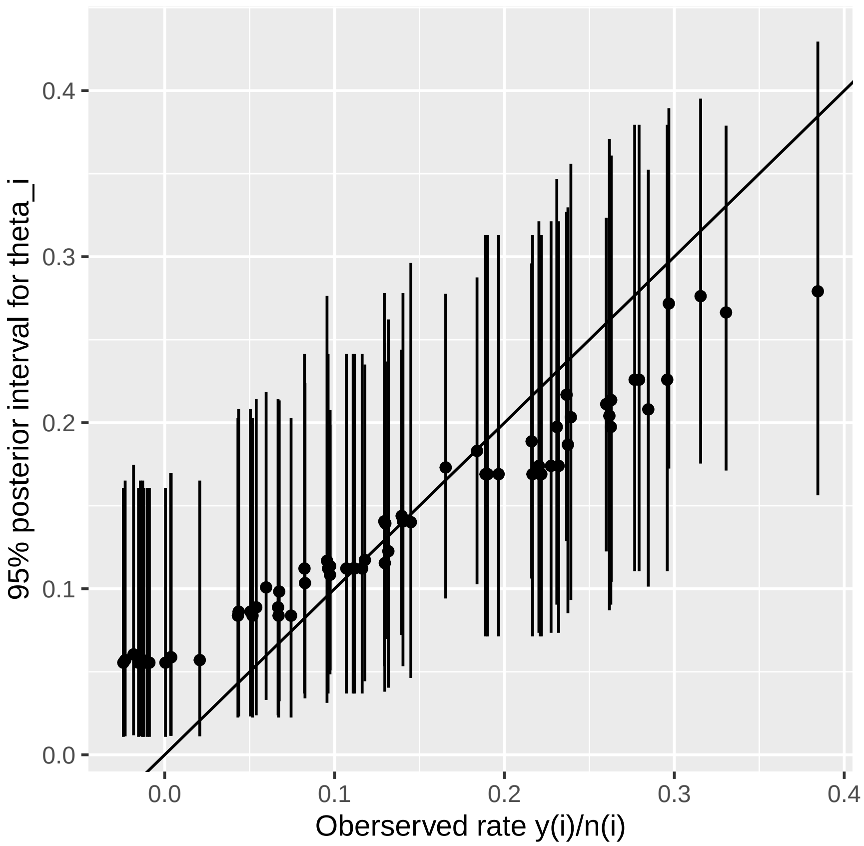


figure 5.4

Here, we sample each θ_j from their respective posterior distribution and compute the 95% confidence interval for these θ using the simulated $\alpha's$ and $\beta's$ from above. We then plot these with the medians shown as points. We also apply a small "jitter" as recommended in Gelman BA to all the x coordinates to reduce overlap.

```
samps <- samps %>% mutate(a=par.alpha(x,y),
         b=par.beta(x,y))

a <- samps$a
b <- samps$b

qh <- function(q, n, y) colMeans(mapply(function(q, n, y, a, b)
    mapply(qbeta, q, a + y, n - y + b), q, n, y, MoreArgs = list(samps$a, samps$b)))


qq_hier <- data.frame(id=1:length(n), n=n, y=y,
                      low=qh(0.025, n, y), high=qh(0.975, n, y), med=qh(0.5, n, y), perc=(y/n))

qq_hier$perc <- jitter(qq_hier$perc, amount=0.025)


ggplot(qq_hier, aes(x=perc, y=med, ymin=low, ymax=high)) +
  geom_point() + geom_abline(intercept=0, slope=1) + geom_linerange() +
  theme(aspect.ratio=1) +  labs(x = "Oberserved rate y(i)/n(i)", y = "95% posterior interval for theta_i
```




```
# qq_hier[order(qq_hier$perc, decreasing = TRUE),]
```

```
# cal.expvals = function(dens) {
#     normPost = dens$logPost - max(dens$logPost)
#     alpha = sum(dens$alpha * exp(normPost)) / sum(exp(normPost))
#     beta = sum(dens$beta * exp(normPost)) / sum(exp(normPost))
#     x = log(alpha / beta)
#     y = log(alpha + beta)
#     mean = alpha / (alpha + beta)
#     data.frame(alpha=alpha, beta=beta, x=x, y=y, mean=mean)
# }
```

Expected values for α, β, x, y, and θ based on pointwise estimates of α and β

```
# grid
#
# exp.vals <- cal.expvals(grid)
#
# exp.alpha <- exp.vals$alpha
# exp.beta <- exp.vals$beta
#
# exp.vals
```

Stan

For our hyperprior distributions, I chose following the justification in this **blog post** (as well as the method outlined on beta distribution's wiki), to reparameterize the the hyper prior in terms of the mean and scale (sample size) of the beta distribution. For the mean, I choose a $unif[0,1]$ distribution, and for the scale, any distribution defined on $[k,\infty]$ for some small k. The pareto distribution seems to be a common choice for such a distribution.

When we do this reparameterize, we get $\phi = \frac{\alpha}{\alpha+\beta}$ and $\lambda = \alpha + \beta$

Then

$$\alpha = \lambda \cdot \phi$$
$$\beta = \lambda \cdot (1 - \phi)$$

```
writeLines(readLines("test1.stan"))
```

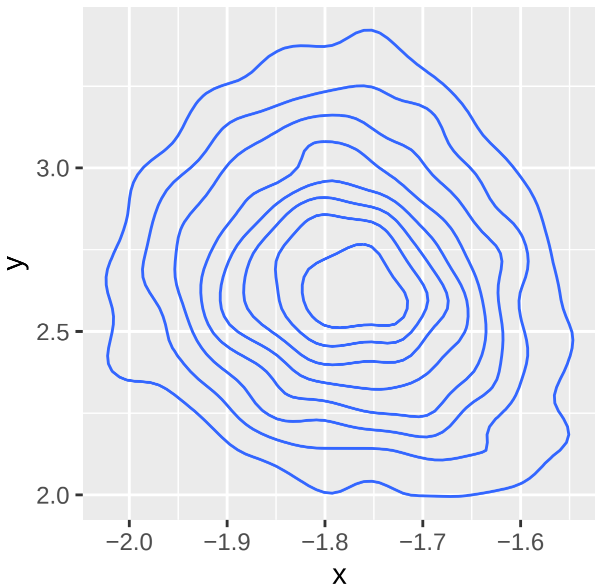
```
## data {
##     int<lower=0> J;
##     int<lower=0> n[J];
##     int<lower=0> y[J];
## }
## parameters {
##     real<lower=0,upper=1> phi;
##     real<lower=0.1> lambda;
##     real<lower=0,upper=1> theta[J];
## }
## transformed parameters {
##     real<lower=0> alpha;
##     real<lower=0> beta;
##     alpha = lambda * phi;
##     beta = lambda * (1 - phi);
## }
## model {
##     phi ~ beta(1,1);
##     lambda ~ pareto(0.1, 1.5);
##     theta ~ beta(alpha, beta);
##     y ~ binomial(n, theta);
## }
```

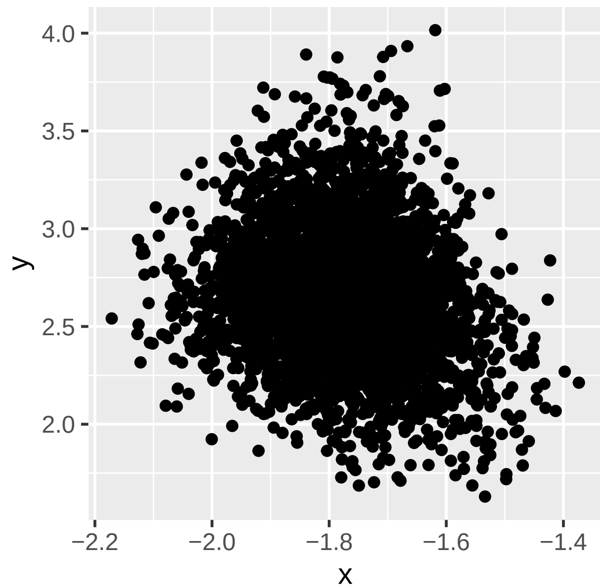
Here we fit the model

```
rat_fit = stan(file="test1.stan", data=list(J=j, y=y, n=n),
               iter=2000, chains=4)

rat_sim = rstan::extract(rat_fit)
# print(rat_fit)
```

below we plot the contour and scatter plot of the simulated posterior distribution as in figures 5.3a and 5.3b.

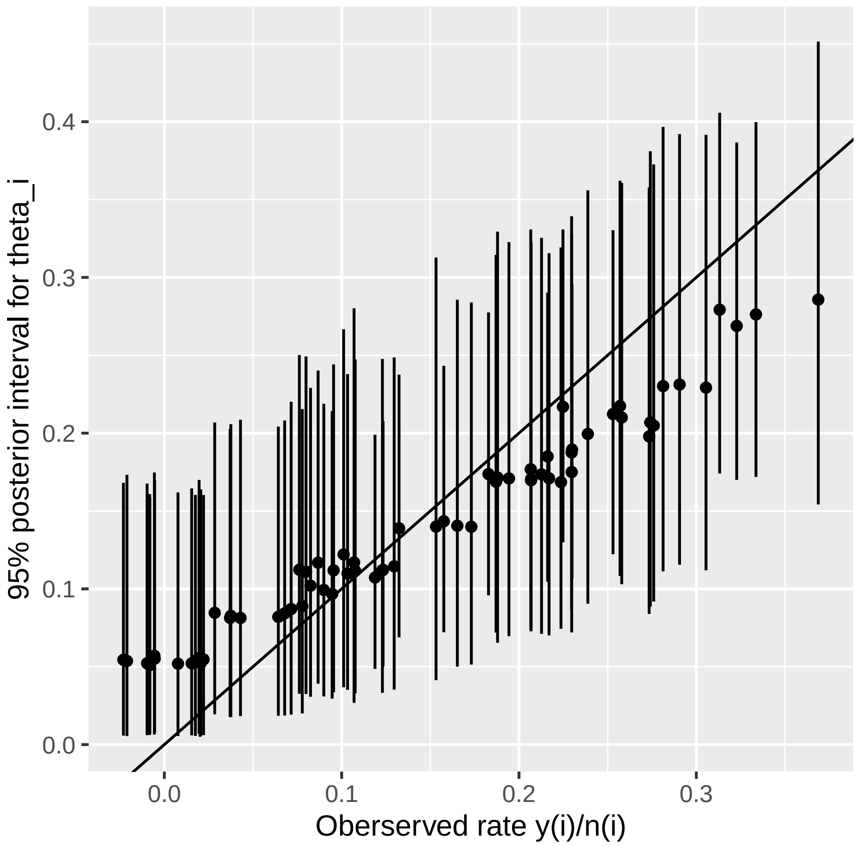
```
a <- rat_sim$alpha
b <- rat_sim$beta

contour <- ggplot(data.frame(x=log(a/b), y=log(a+b), a=a, b=b)) +
  # geom_point(aes(x=x, y=y)) +
  geom_density_2d(aes(x=x, y=y)) + theme(aspect.ratio=1)

plot <- ggplot(data.frame(x=log(a/b), y=log(a+b), a=a, b=b)) +
  geom_point(aes(x=x, y=y)) + theme(aspect.ratio=1)
  # geom_density_2d(aes(x=x, y=y))

grid.arrange(contour, plot, nrow = 1)
```





And here we plot the median's and 95% confidence intervals for each θ_i) as in figure 5.4

```
theta_summary <- summary(rat_fit, pars = c("theta"), probs = c(0.025, 0.5, 0.975))$summary

qq_hier <- data.frame(id=1:length(n), n=n, y=y,
                      low=theta_summary[,"2.5%"], high=theta_summary[,"97.5%"], med=theta_summary[,"50%

qq_hier$perc <- jitter(qq_hier$perc, amount=0.025)


ggplot(qq_hier, aes(x=perc, y=med, ymin=low, ymax=high)) +
  geom_point() + geom_abline(intercept=0, slope=1) + geom_linerange() +
  theme(aspect.ratio=1) +  labs(x = "Oberserved rate y(i)/n(i)", y = "95% posterior interval for theta_
```


95% posterior interval for theta_i
0.0
0.1
0.2
0.3
0.4
0.0
0.1
0.2
0.3
Oberserved rate y(i)/n(i)