```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(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)
```

```r
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**

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

, 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) }
```

Below are the point-wise estimates on the grid.

figure 5.2

```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
```

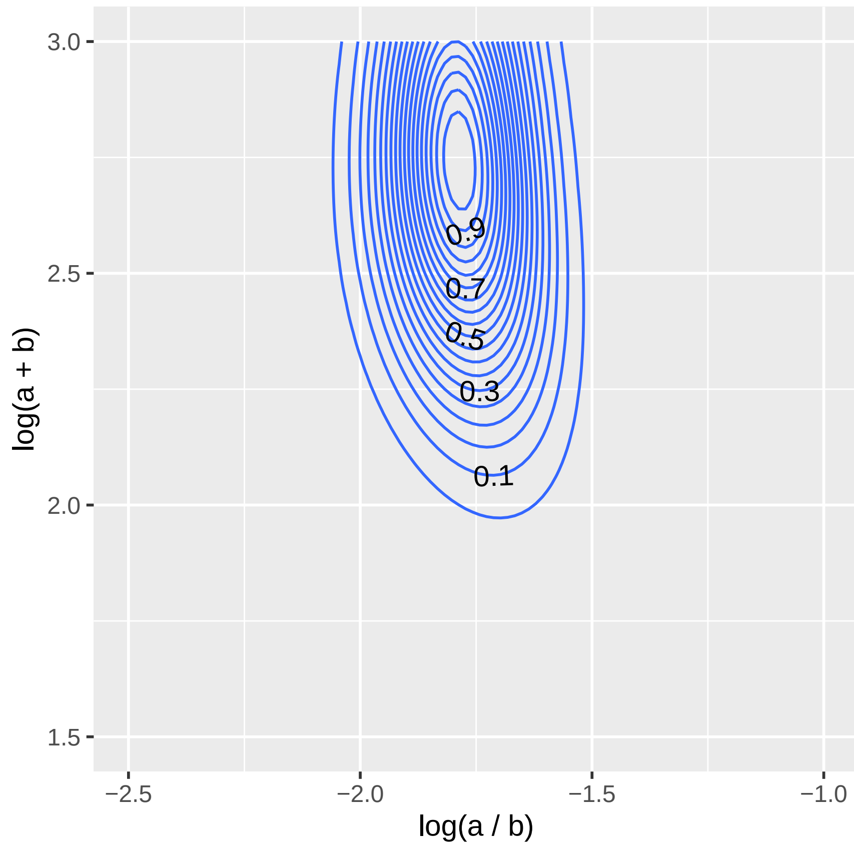



**figure 5.3a**

```
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
```

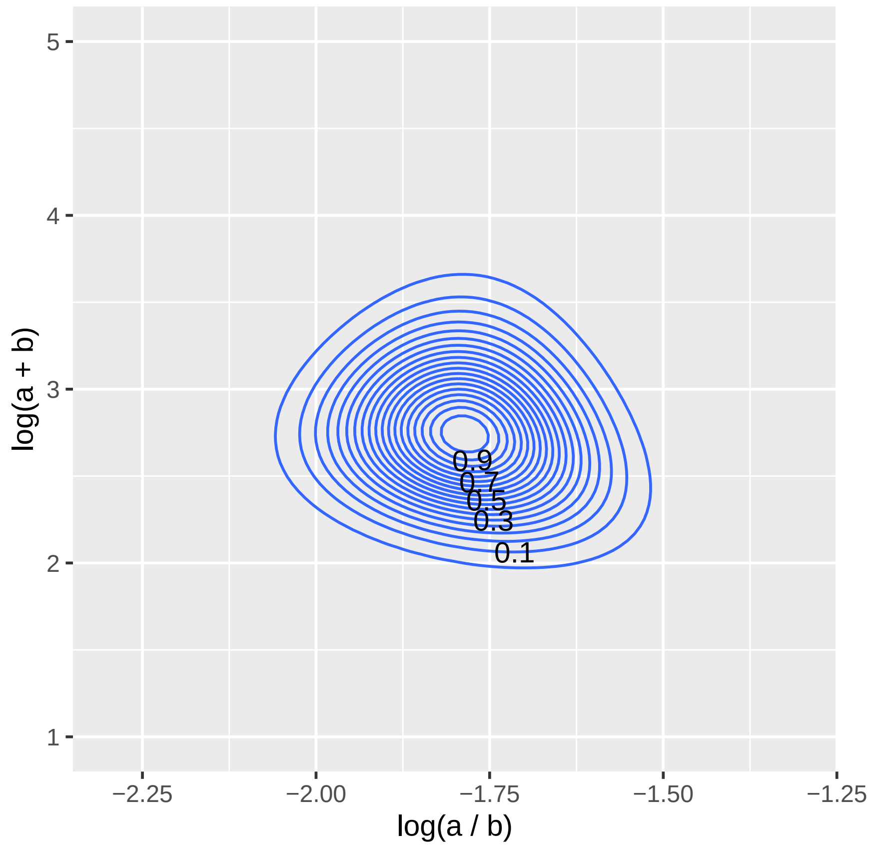



**figure 5.3b**

```
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
```

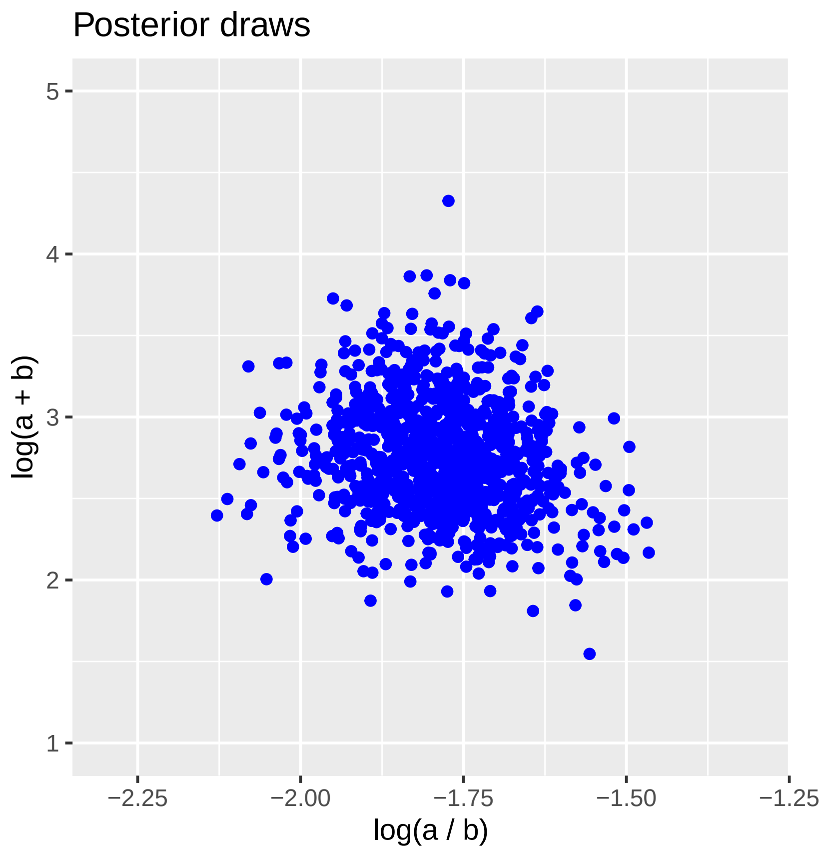
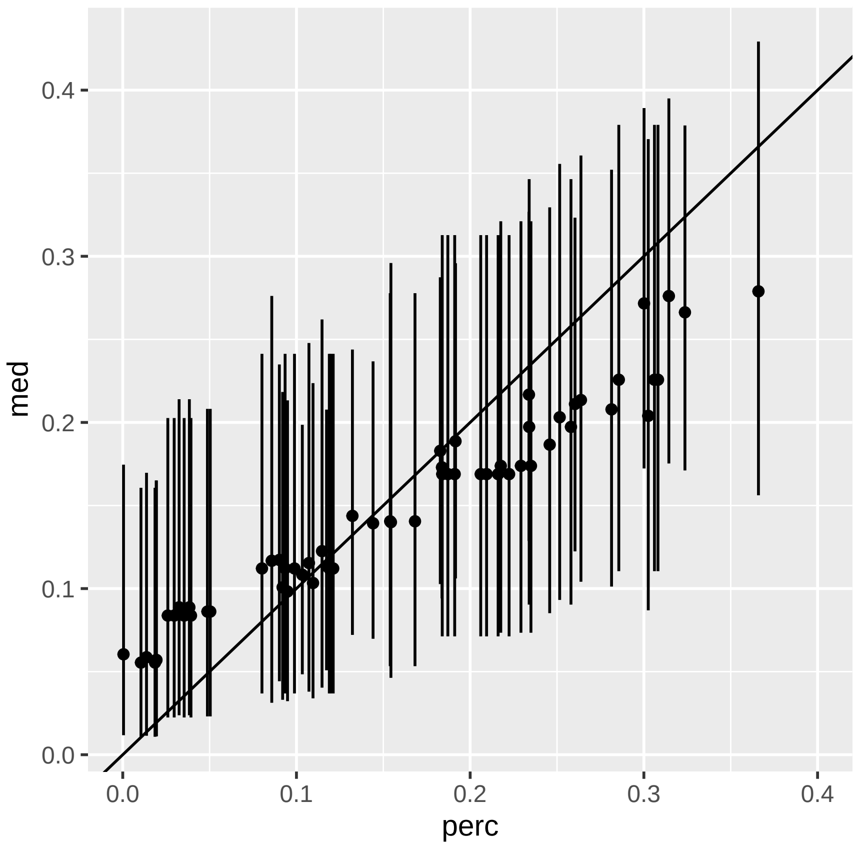



**figure 5.4**

```
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) + xlim(0, 0.4)
```

```
## Warning: Removed 8 rows containing missing values (geom_point).

## Warning: Removed 8 rows containing missing values (geom_segment).
```




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

```
##    id  n  y        low      high        med         perc
## 70 70 24  9 0.15615684 0.4292570 0.27891081  0.3659762629
## 67 67 52 16 0.17110388 0.3787494 0.26626777  0.3236457951
## 68 68 46 15 0.17530082 0.3950122 0.27607321  0.3143949961
## 64 64 20  6 0.11048081 0.3791038 0.22570801  0.3081296965
## 65 65 20  6 0.11048081 0.3791038 0.22570801  0.3061006961
## 71 71 14  4 0.08690504 0.3705163 0.20397731  0.3025326392
## 69 69 47 15 0.17230310 0.3892202 0.27164176  0.3001088559
## 66 66 20  6 0.11048081 0.3791038 0.22570801  0.2855751282
## 61 61 23  6 0.10121422 0.3520960 0.20786312  0.2813904033
## 63 63 22  6 0.10412100 0.3606597 0.21348311  0.2637648445
## 57 57 46 11 0.12239679 0.3232490 0.21108213  0.2603508015
## 60 60 20  5 0.09035279 0.3464345 0.19730089  0.2580299655
## 62 62 19  5 0.09317470 0.3555981 0.20311562  0.2515345886
## 56 56 22  5 0.08520400 0.3294616 0.18663459  0.2457818283
## 54 54 19  4 0.07342712 0.3210695 0.17386157  0.2349886465
## 59 59 20  5 0.09035279 0.3464345 0.19730089  0.2339708660
## 58 58 49 12 0.12866641 0.3267714 0.21673436  0.2338473870
## 55 55 19  4 0.07342712 0.3210695 0.17386157  0.2292296521
## 45 45 20  4 0.07123063 0.3127345 0.16889739  0.2224115210
## 53 53 19  4 0.07342712 0.3210695 0.17386157  0.2176285546
## 48 48 20  4 0.07123063 0.3127345 0.16889739  0.2161080622
## 46 46 20  4 0.07123063 0.3127345 0.16889739  0.2094526772
## 47 47 20  4 0.07123063 0.3127345 0.16889739  0.2060805376
## 52 52 48 10 0.10609232 0.2958089 0.18868985  0.1915711799
```

```
## 51 51 20  4 0.07123063 0.3127345 0.16889739  0.1910926813
## 49 49 20  4 0.07123063 0.3127345 0.16889739  0.1871465559
## 50 50 20  4 0.07123063 0.3127345 0.16889739  0.1839278109
## 43 43 48  9 0.09409384 0.2775814 0.17295839  0.1837738739
## 44 44 50 10 0.10270037 0.2873717 0.18292319  0.1827892273
## 40 40 20  3 0.05330725 0.2777946 0.14049988  0.1682398429
## 42 42 13  2 0.04629196 0.2959628 0.13996632  0.1543820281
## 41 41 20  3 0.05330725 0.2777946 0.14049988  0.1539704267
## 38 38 49  7 0.06980652 0.2367598 0.13930519  0.1441090524
## 39 39 47  7 0.07212265 0.2438795 0.14375962  0.1322163182
## 27 27 20  2 0.03687603 0.2412872 0.11211334  0.1211523507
## 28 28 20  2 0.03687603 0.2412872 0.11211334  0.1197140982
## 30 30 20  2 0.03687603 0.2412872 0.11211334  0.1189179692
## 35 35 46  5 0.05083801 0.2077193 0.11360828  0.1173158448
## 37 37 17  2 0.04038184 0.2619897 0.12252654  0.1147210536
## 25 25 23  2 0.03394274 0.2236553 0.10336337  0.1095344424
## 34 34 19  2 0.03797306 0.2478090 0.11537711  0.1071814734
## 33 33 49  5 0.04841344 0.1985694 0.10833859  0.1033715182
## 29 29 20  2 0.03687603 0.2412872 0.11211334  0.0988657635
## 23 23 25  2 0.03223845 0.2132821 0.09826431  0.0948535606
## 26 26 20  2 0.03687603 0.2412872 0.11211334  0.0934509522
## 24 24 24  2 0.03306817 0.2183440 0.10074813  0.0920392906
## 36 36 27  3 0.04423135 0.2349020 0.11725484  0.0901634219
## 32 32 10  1 0.03126870 0.2761402 0.11674035  0.0857787299
## 31 31 20  2 0.03687603 0.2412872 0.11211334  0.0801378809
## 20 20 19  1 0.02306527 0.2081528 0.08616503  0.0503777889
## 19 19 19  1 0.02306527 0.2081528 0.08616503  0.0487401750
## 16 16 20  1 0.02241869 0.2026432 0.08375054  0.0392697280
## 21 21 18  1 0.02375142 0.2139762 0.08872669  0.0383452193
## 18 18 20  1 0.02241869 0.2026432 0.08375054  0.0353535193
## 22 22 18  1 0.02375142 0.2139762 0.08872669  0.0324465897
## 15 15 20  1 0.02241869 0.2026432 0.08375054  0.0296035281
## 17 17 20  1 0.02241869 0.2026432 0.08375054  0.0259344608
## 11 11 19  0 0.01110552 0.1650524 0.05702600  0.0193898973
## 8   8 19  0 0.01110552 0.1650524 0.05702600  0.0192529653
## 2   2 20  0 0.01081104 0.1606690 0.05545847  0.0185773292
## 12 12 18  0 0.01141701 0.1696872 0.05868727  0.0136693154
## 4   4 20  0 0.01081104 0.1606690 0.05545847  0.0104809715
## 14 14 17  0 0.01174709 0.1745961 0.06045116  0.0004411589
## 5   5 20  0 0.01081104 0.1606690 0.05545847 -0.0095330877
## 6   6 20  0 0.01081104 0.1606690 0.05545847 -0.0112446278
## 3   3 20  0 0.01081104 0.1606690 0.05545847 -0.0186027431
## 9   9 19  0 0.01110552 0.1650524 0.05702600 -0.0186840172
## 10 10 19  0 0.01110552 0.1650524 0.05702600 -0.0192938454
## 13 13 18  0 0.01141701 0.1696872 0.05868727 -0.0193634361
## 1   1 20  0 0.01081104 0.1606690 0.05545847 -0.0220645899
## 7   7 20  0 0.01081104 0.1606690 0.05545847 -0.0244220243
```

```r
# 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 $\alpha, \beta, x, y$, and $\theta$ based on pointwise estimates of $\alpha$ and $\beta$

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

### Stan

Fit the model

```
writeLines(readLines("ratmodel.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,upper=1> theta[J];
## }
## transformed parameters {
##     real<lower=0> alpha;
##     real<lower=0> beta;
## }
## model {
##     phi ~ beta(1,1);
##     theta ~ beta(alpha, beta);
##     y ~ binomial(n, theta);
## }
```

```
rat_fit = stan(file="ratmodel-old.stan", data=list(J=j, y=y, n=n),
               iter=2000, chains=4, control=list(adapt_delta=0.99))
```

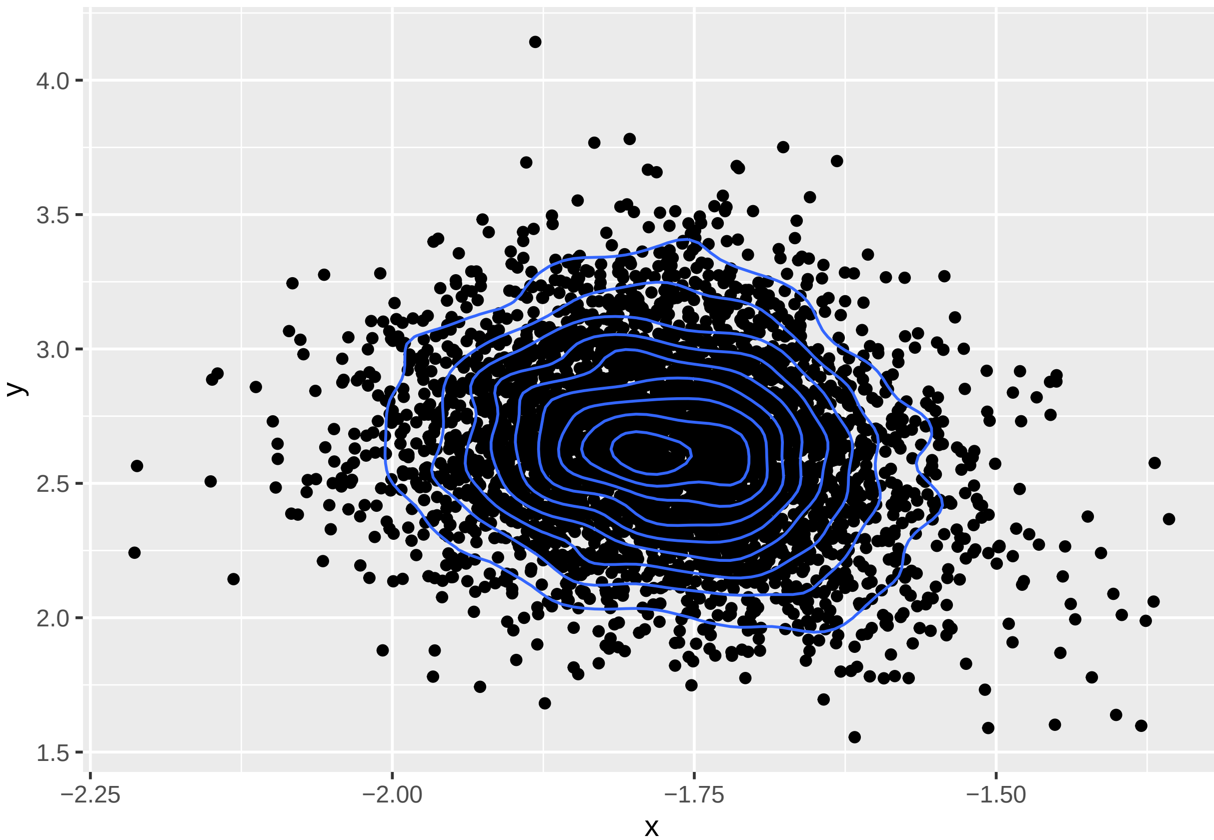
```
# pairs(rat_fit, pars=c("alpha", "beta", "lp__"))
rat_sim = rstan::extract(rat_fit, permuted=TRUE)

n_sims = length(rat_sim$lp__)
n_sims
```

```
## [1] 4000
```

Simulated contour and points in figure 5.3(a and b)

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




```
# contour(kde2d(log(a/b), log(a+b)))
```

```
theta_sims = data.frame(alpha=rat_sim$alpha, beta=rat_sim$beta) %>%
    mutate(Theta=rbeta(n(), alpha+y[1], beta + n[1] - y[1]))
```