```
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()
```

```
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
```

```
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
```

```
registerDoParallel()


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

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

```
tumor_experiments$percent = tumor_experiments$tumors / tumor_experiments$n
# y <- tumor_experiments$tumors
# n <- tumor_experiments$n
y <- c(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

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

, We will compute this log-transformed.

```
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.

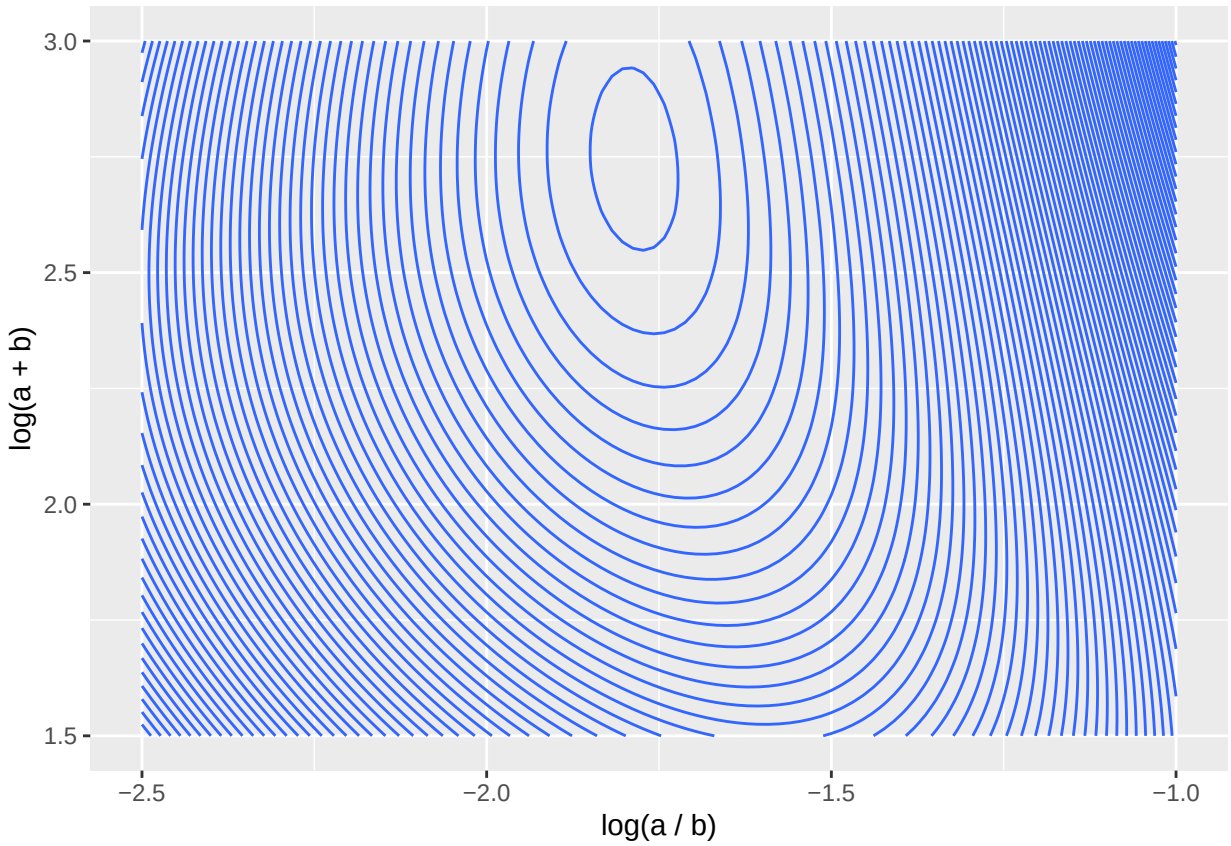
figure 5.2

```
xcord <- seq(-2.5, -1, length.out = 100)
ycord <- seq(1.5, 3, 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))



ggplot(grid) +
    geom_contour(aes(x=x, y=y, z=z), binwidth=0.5)  +
    xlab("log(a / b)") +
    ylab("log(a + b)")
```



unused

```
# model.loglike = function(alpha, beta, cs, ns) {
#     base = lgamma(alpha + beta) - lgamma(alpha) - lgamma(beta)
#     data = mapply(a=alpha, b=beta, FUN=function(a, b) {
#         sum(lgamma(a + cs) +
#                 lgamma(b + ns - cs) -
#                 lgamma(a + b + ns))
#     })
#     nrow(tumor_experiments) * base + data
# }
# xydensity = function(x, y, counts, totals, prior=model.prior, loglike=model.loglike) {
#     expand.grid(x=x, y=y) %>%
#         mutate(alpha = par.alpha(x, y),
#                beta = par.beta(x, y)) %>%
#         mutate(logPrior = log(prior(alpha, beta)),
#                logLike = loglike(alpha, beta, counts, totals),
#                rawLogPost = logPrior + logLike) %>%
#         mutate(logJacobian = log(alpha) + log(beta),
#                logPost = rawLogPost + logJacobian)
# }
```

```
# xcord <- seq(-2.5, -1, length.out = 100)
# ycord <- seq(1.5, 3, length.out = 100)
#
# # xcord <- seq(-2.3, -1.3, length.out = 100)
# # ycord <- seq(1, 5, length.out = 100)
# dens.points = xydensity(x=xcord,
```

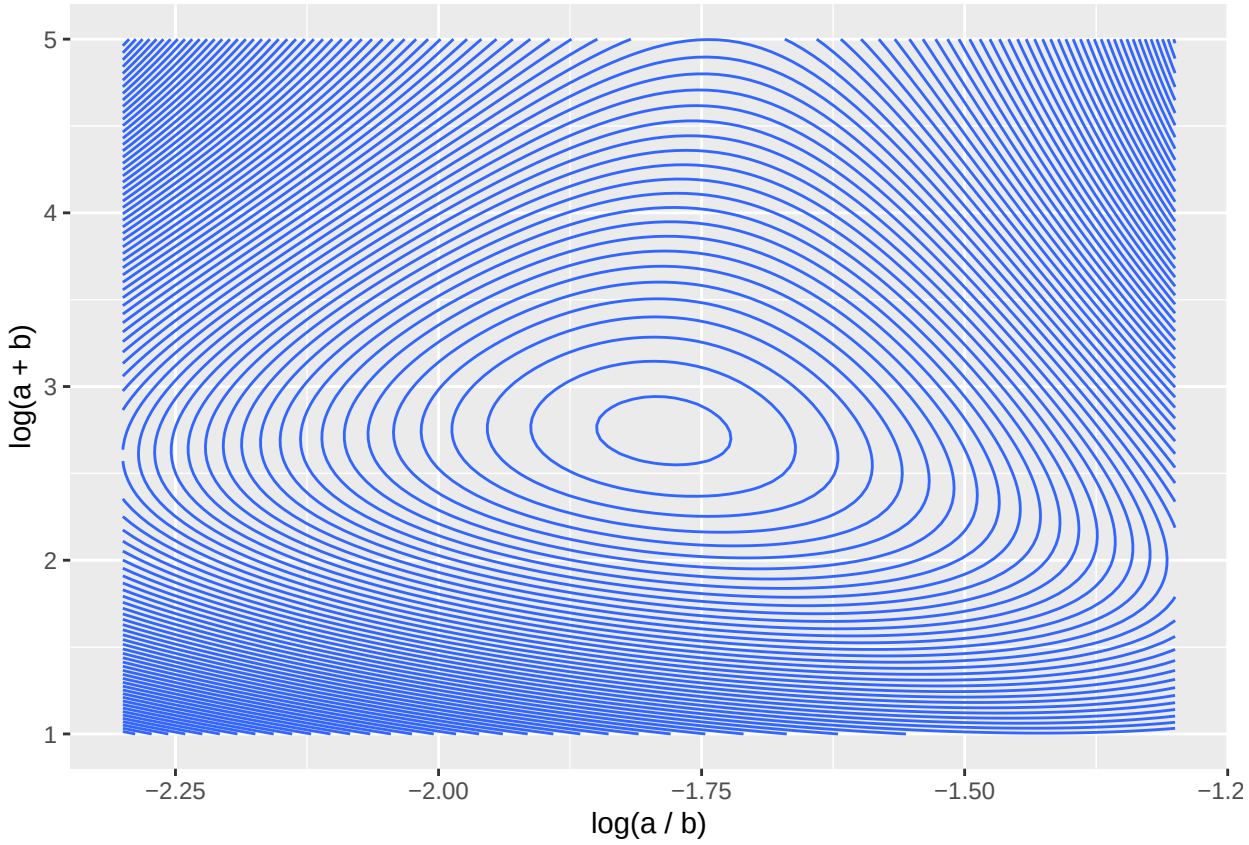
```
#                         y=ycord,
#                         counts=y, totals=n)
# ggplot(dens.points) +
#     geom_contour(aes(x=x, y=y, z=logPost), binwidth=0.5, geom="contour")  +
#     xlab("log(a / b)") +
#     ylab("log(a + b)")
```

figure 5.3

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

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


ggplot(grid) +
    geom_contour(aes(x=x, y=y, z=logPost), binwidth=0.5)  +
    xlab("log(a / b)") +
    ylab("log(a + b)")
```




```
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

```
exp.vals <- cal.expvals(grid)

exp.alpha <- exp.vals$alpha
exp.beta <- exp.vals$beta

exp.vals
```

```
##      alpha    beta         x       y      mean
## 1 2.402568 14.3195 -1.785085 2.81673 0.1436764
```

```
# theta.post.point = data.frame(Theta=seq(0,1,0.001))
# theta.post.point$PostDens = dbeta(theta.post.point$Theta, exp.alpha, exp.beta)
# ggplot(theta.post.point) + aes(x=Theta, y=PostDens) + geom_line()
```

```
# model.map = optim(c(1, 1), function(pars) {
#     alpha = pars[1]
#     beta = pars[2]
#     log(model.prior(alpha, beta)) + model.loglike(alpha, beta, y, n)
# }, control=list(fnscale=-1))
# model.map
```

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=0.1> 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);
```

```
## }
```

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

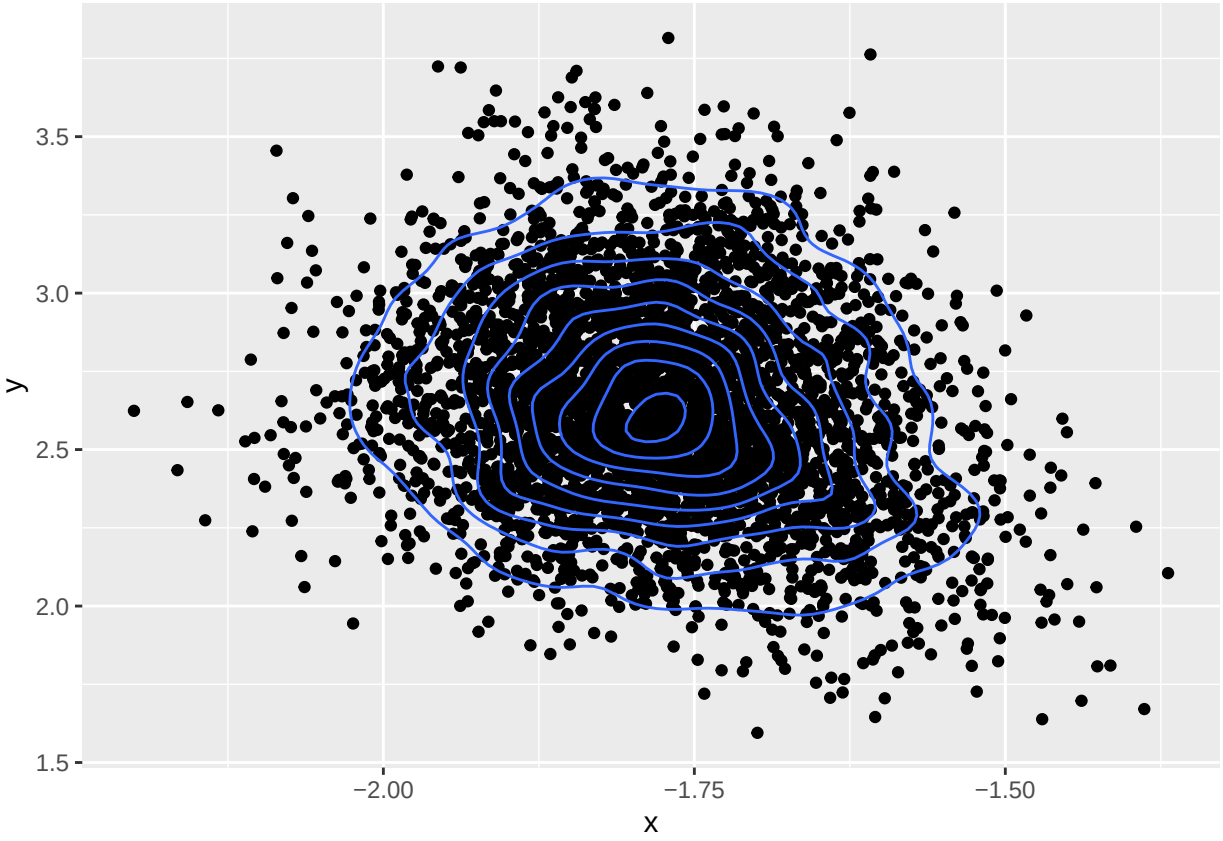
```
# pairs(rat_fit, pars=c("alpha", "beta", "lp__"))
rat_sim = rstan::extract(rat_fit, permuted=TRUE)

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]))
```