Problem Sets Between Tests 1 and 2

*Only turn in problems that are **not** bracketed.* Bracketed problems are additional problems you can look at. Round brackets indicate problems that may help you with problems that are assigned; square brackets are additional problems on material that you should know, but you are not required to write up solutions; curly brackets are truly optional and may contain extra nuggets that you will not be required to know but may be interested in.

Additional assignments will be filled in over time.

notation	meaning
unbracketed	assigned problem – turn these in for grading
()	helper/warm-up problem
[]	additional problems (you are responsible for content, but don't turn them in)
{}	covers optional material

PS	Due	Source	Problems
9	Wed 3/8	Rethinking 6 Additional Problems	**6E3–6E4** entropy **1** entropy **2** entropy
10	Fri 3/17	Rethinking 6	**6M3** same data **6M4** narrow prior **6H1** compare models **6H4** train vs. test **6H5** train vs. test *Use the replacement code below in place of the R code 6.31 and 6.32.*
11	Wed 3/29	Rethinking 6	**6M2** selection v averaging **6M5–6M6** over/under fitting **6H2** plotting models **6H3** model averaging
12	Wed 4/5	Rethinking 7 Rethinking 8	**7H3** loo **8H3 – 8H4** Note: You can find replacement code for the code the author provides in the usual place online.
13	Thu 4/13	Rethinking 10	**10E1 – 10E2** warm-up **10M1** likelihood **10M3** link **10H1** map vs stan **10H2** chimp models **10H3** eagle pirates
14	Fri 4/21	Rethinking 12	**12E1** shrinkage **12E2** multi-level **12E3** multi-level
15	Mon 4/24	Rethinking 12	**12M1** tadpoles **12M2** tadpoles **12M3** priors **12H1** contraception Note: For problem 12M3, rather than have you all print out the model fitting and `precis()` code, I've included it below. You don't need to include it in your print out. Note: `coeftab()` is useful for comparing coefficients across different models. I've updated the `rethinking` package so you can use square brackets to grab just some rows/columns to save some paper. Also, be sure to include `require(rstan)` as well as `require(rethinking)` now.

Replacement Code

12M3

This isn't really replacement code, I'm just going to save you some time by giving you code and output that you don't need to repeat in the work you turn in. You can run the code and do other things if you like. But don't include the code that creates these models or the precis output in your homework. Let's save a tree.

[Note: an earlier version of this only did the Cauchy model because I thought that this output for the Gaussian (ie, normal) model was in the book. Turns out I remembered that incorrectly. So now the output for both models is here for you.]

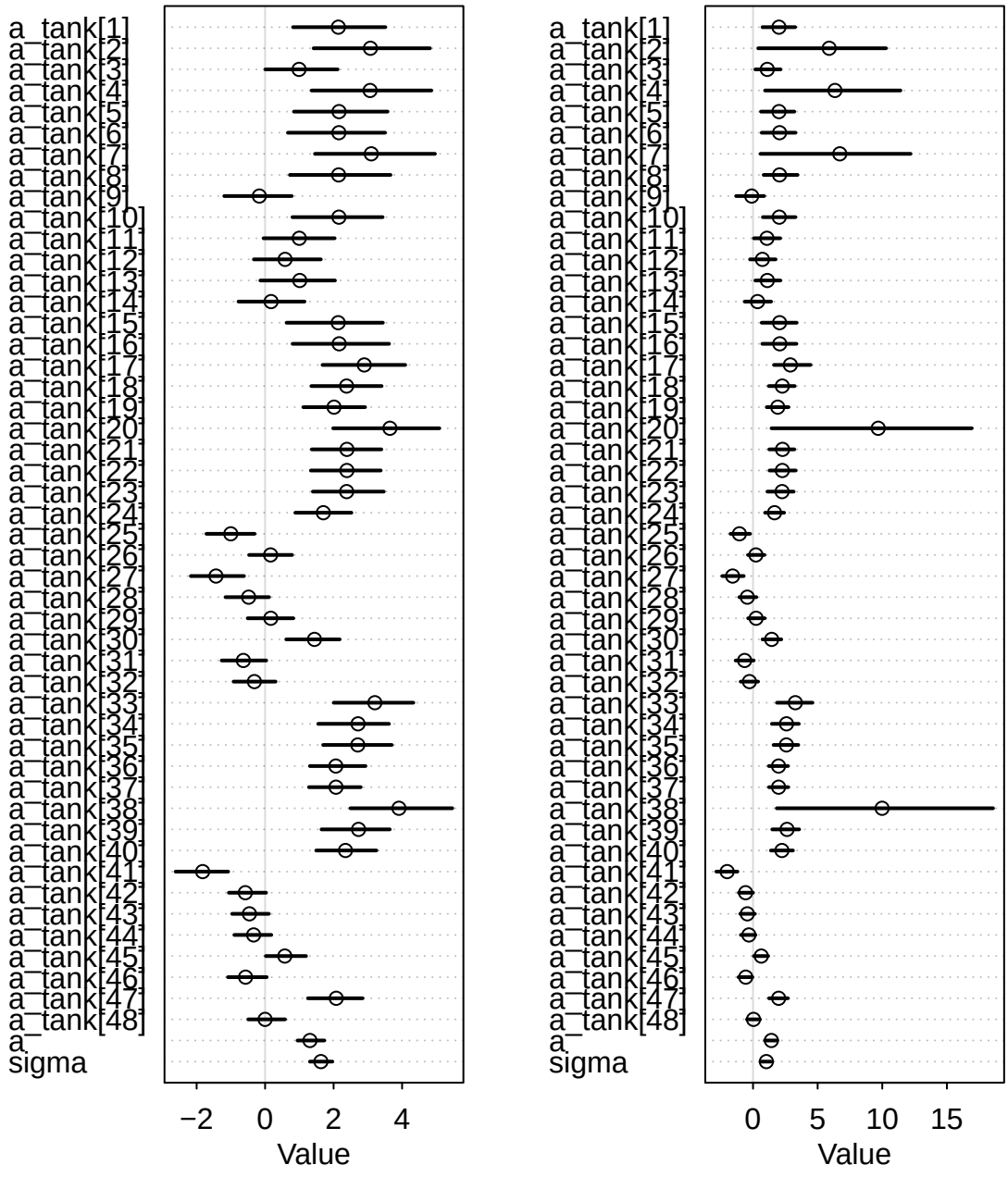
```
data(reedfrogs)
Frogs <- reedfrogs %>% mutate(tank = 1:n())   # make the tank cluster variable
m.12h3g <- map2stan(
  alist(
    surv ~ dbinom(density, p),
    logit(p) <- a_tank[tank],
    a_tank[tank] ~ dnorm(a, sigma),
    a ~ dnorm(0, 1),
    sigma ~ dcauchy(0, 1)
  ),
  data = Frogs, refresh = 0, iter = 4000
)

##
##  Elapsed Time: 0.341924 seconds (Warm-up)
##                0.229005 seconds (Sampling)
##                0.570929 seconds (Total)

## The following numerical problems occurred the indicated number of times on chain 1
##                                                                                 count
## Exception thrown at line 17:  normal_log:  Scale parameter is 0, but must be > 0!    1
## When a numerical problem occurs, the Hamiltonian proposal gets rejected.
## See http://mc-stan.org/misc/warnings.html#exception-hamiltonian-proposal-rejected
## If the number in the 'count' column is small, there is no need to ask about this message on stan-users.

##
## SAMPLING FOR MODEL 'surv ~ dbinom(density, p)' NOW (CHAIN 1).
## WARNING: No variance estimation is
##          performed for num_warmup < 20
##
##
## Chain 1, Iteration: 1 / 1 [100%]  (Sampling)
##  Elapsed Time: 3e-06 seconds (Warm-up)
##                7e-05 seconds (Sampling)
##                7.3e-05 seconds (Total)

## Computing WAIC
## Constructing posterior predictions

## [ 200 / 2000 ]
[ 400 / 2000 ]
[ 600 / 2000 ]
[ 800 / 2000 ]
[ 1000 / 2000 ]
[ 1200 / 2000 ]
[ 1400 / 2000 ]
[ 1600 / 2000 ]
[ 1800 / 2000 ]
[ 2000 / 2000 ]
```

```
## Aggregated binomial counts detected.  Splitting to 0/1 outcome for WAIC calculation.

m.12h3c <- map2stan(
  alist(
    surv ~ dbinom(density, p),
    logit(p) <- a_tank[tank],
    a_tank[tank] ~ dcauchy(a, sigma),
    a ~ dnorm(0, 1),
    sigma ~ dcauchy(0, 1)
  ),
  data = Frogs, refresh = 0, iter = 4000
)

##
##  Elapsed Time: 2.02369 seconds (Warm-up)
##                10.2402 seconds (Sampling)
##                12.2638 seconds (Total)
##
##
## SAMPLING FOR MODEL 'surv ~ dbinom(density, p)' NOW (CHAIN 1).
## WARNING: No variance estimation is
##          performed for num_warmup < 20
##
##
## Chain 1, Iteration: 1 / 1 [100%]  (Sampling)
##  Elapsed Time: 2e-06 seconds (Warm-up)
##                9.9e-05 seconds (Sampling)
##                0.000101 seconds (Total)

## Computing WAIC
## Constructing posterior predictions

## [ 200 / 2000 ]
[ 400 / 2000 ]
[ 600 / 2000 ]
[ 800 / 2000 ]
[ 1000 / 2000 ]
[ 1200 / 2000 ]
[ 1400 / 2000 ]
[ 1600 / 2000 ]
[ 1800 / 2000 ]
[ 2000 / 2000 ]

## Aggregated binomial counts detected.  Splitting to 0/1 outcome for WAIC calculation.
```

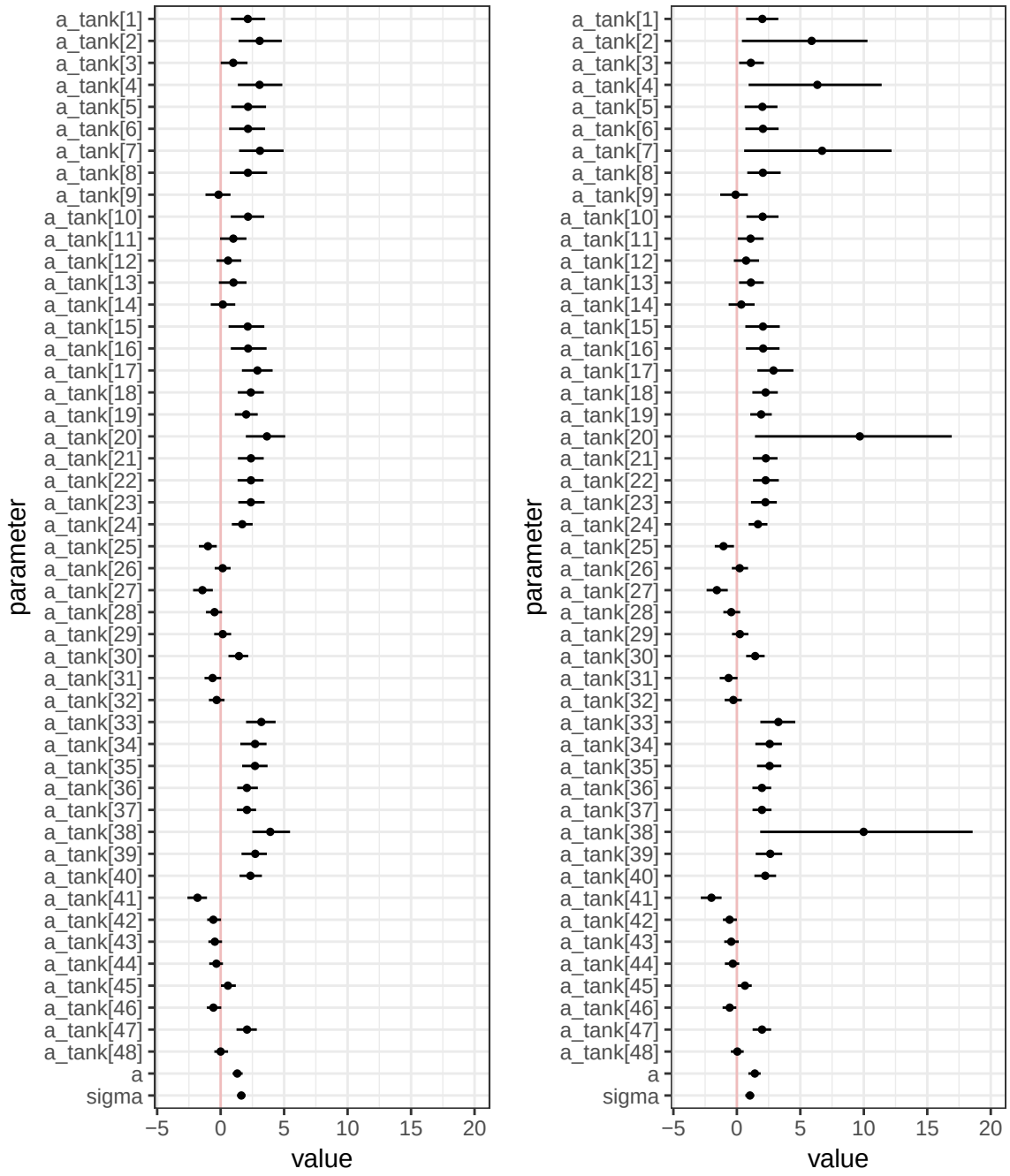
```
precis(m.12h3g)  # only the highest level parameters

## 48 vector or matrix parameters omitted in display.  Use depth=2 to show them.

##       Mean StdDev lower 0.89 upper 0.89 n_eff Rhat
## a     1.31   0.26       0.95       1.73  2000    1
## sigma 1.63   0.21       1.31       1.97  1073    1

precis(m.12h3c)  # only the highest level parameters

## 48 vector or matrix parameters omitted in display.  Use depth=2 to show them.

##       Mean StdDev lower 0.89 upper 0.89 n_eff Rhat
## a     1.42   0.30       0.91       1.88  1413    1
## sigma 1.03   0.23       0.66       1.37  1325    1
```

```
precis(m.12h3g, depth = 2)      # all the parameters

##             Mean StdDev lower 0.89 upper 0.89 n_eff Rhat
## a_tank[1]   2.14   0.87       0.81       3.51  2000    1
## a_tank[2]   3.08   1.11       1.42       4.81  2000    1
## a_tank[3]   0.99   0.67       0.00       2.12  2000    1
## a_tank[4]   3.07   1.15       1.36       4.86  2000    1
## a_tank[5]   2.16   0.89       0.84       3.58  2000    1
## a_tank[6]   2.15   0.91       0.67       3.50  2000    1
## a_tank[7]   3.10   1.11       1.45       4.96  1409    1
## a_tank[8]   2.15   0.94       0.72       3.66  2000    1
## a_tank[9]  -0.17   0.61      -1.19       0.78  2000    1
## a_tank[10]  2.15   0.85       0.80       3.43  2000    1
## a_tank[11]  1.00   0.67      -0.05       2.03  2000    1
## a_tank[12]  0.58   0.62      -0.33       1.63  2000    1
## a_tank[13]  1.01   0.69      -0.14       2.04  2000    1
## a_tank[14]  0.17   0.61      -0.78       1.15  2000    1
## a_tank[15]  2.13   0.90       0.63       3.44  2000    1
## a_tank[16]  2.16   0.90       0.80       3.62  2000    1
## a_tank[17]  2.90   0.78       1.68       4.09  2000    1
## a_tank[18]  2.38   0.66       1.35       3.40  2000    1
## a_tank[19]  2.01   0.57       1.11       2.92  2000    1
## a_tank[20]  3.64   1.03       1.99       5.09  2000    1
## a_tank[21]  2.39   0.66       1.36       3.39  2000    1
## a_tank[22]  2.39   0.67       1.34       3.38  2000    1
## a_tank[23]  2.38   0.68       1.39       3.47  2000    1
## a_tank[24]  1.70   0.53       0.88       2.52  2000    1
## a_tank[25] -1.00   0.44      -1.71      -0.31  2000    1
## a_tank[26]  0.16   0.40      -0.47       0.79  2000    1
## a_tank[27] -1.43   0.50      -2.17      -0.62  2000    1
## a_tank[28] -0.48   0.40      -1.16       0.11  2000    1
## a_tank[29]  0.17   0.42      -0.51       0.82  2000    1
## a_tank[30]  1.44   0.50       0.62       2.18  2000    1
## a_tank[31] -0.64   0.41      -1.27       0.03  2000    1
## a_tank[32] -0.31   0.40      -0.92       0.31  2000    1
## a_tank[33]  3.20   0.74       2.00       4.33  2000    1
## a_tank[34]  2.71   0.66       1.55       3.62  2000    1
## a_tank[35]  2.71   0.66       1.69       3.70  2000    1
## a_tank[36]  2.07   0.52       1.31       2.93  2000    1
## a_tank[37]  2.07   0.50       1.28       2.79  2000    1
## a_tank[38]  3.91   0.98       2.49       5.47  1424    1
## a_tank[39]  2.73   0.65       1.65       3.64  2000    1
## a_tank[40]  2.35   0.56       1.49       3.25  2000    1
## a_tank[41] -1.82   0.49      -2.61      -1.08  2000    1
## a_tank[42] -0.58   0.34      -1.06       0.03  2000    1
## a_tank[43] -0.46   0.34      -0.96       0.11  2000    1
## a_tank[44] -0.34   0.34      -0.90       0.18  2000    1
## a_tank[45]  0.57   0.37       0.02       1.20  2000    1
## a_tank[46] -0.57   0.36      -1.09       0.05  2000    1
## a_tank[47]  2.08   0.51       1.25       2.85  2000    1
## a_tank[48]  0.00   0.34      -0.49       0.58  2000    1
## a           1.31   0.26       0.95       1.73  2000    1
## sigma       1.63   0.21       1.31       1.97  1073    1

precis(m.12h3c, depth = 2)      # all the parameters

##             Mean StdDev lower 0.89 upper 0.89 n_eff Rhat
## a_tank[1]   2.01   0.83       0.73       3.26  2000 1.00
## a_tank[2]   5.89   7.95       0.40      10.29   278 1.00
## a_tank[3]   1.10   0.62       0.18       2.13  2000 1.00
## a_tank[4]   6.34  10.83       0.93      11.41   295 1.01
```

```
## a_tank[5]   2.01   0.89       0.61       3.20  1356 1.00
## a_tank[6]   2.06   0.87       0.67       3.29  1127 1.00
## a_tank[7]   6.71  10.73       0.58      12.19   176 1.01
## a_tank[8]   2.05   0.88       0.82       3.45  1289 1.00
## a_tank[9]  -0.10   0.68      -1.31       0.86  2000 1.00
## a_tank[10]  2.03   0.86       0.76       3.28  1229 1.00
## a_tank[11]  1.08   0.63       0.06       2.11  2000 1.00
## a_tank[12]  0.73   0.63      -0.24       1.75  2000 1.00
## a_tank[13]  1.11   0.63       0.18       2.12  2000 1.00
## a_tank[14]  0.35   0.65      -0.65       1.40  2000 1.00
## a_tank[15]  2.06   0.90       0.67       3.38   819 1.00
## a_tank[16]  2.07   0.93       0.72       3.36   920 1.00
## a_tank[17]  2.89   0.94       1.62       4.46  1173 1.00
## a_tank[18]  2.26   0.65       1.22       3.22  2000 1.00
## a_tank[19]  1.91   0.55       1.06       2.75  2000 1.00
## a_tank[20]  9.69  17.27       1.43      16.93   251 1.01
## a_tank[21]  2.28   0.64       1.26       3.20  1602 1.00
## a_tank[22]  2.27   0.67       1.27       3.30  2000 1.00
## a_tank[23]  2.25   0.67       1.11       3.15  2000 1.00
## a_tank[24]  1.66   0.48       0.93       2.41  2000 1.00
## a_tank[25] -1.05   0.47      -1.74      -0.23  2000 1.00
## a_tank[26]  0.22   0.41      -0.39       0.88  2000 1.00
## a_tank[27] -1.58   0.53      -2.38      -0.72  2000 1.00
## a_tank[28] -0.44   0.42      -1.07       0.26  2000 1.00
## a_tank[29]  0.24   0.41      -0.38       0.91  2000 1.00
## a_tank[30]  1.44   0.46       0.73       2.19  2000 1.00
## a_tank[31] -0.65   0.44      -1.35       0.06  2000 1.00
## a_tank[32] -0.28   0.43      -0.96       0.39  2000 1.00
## a_tank[33]  3.27   0.97       1.85       4.60  1142 1.00
## a_tank[34]  2.59   0.67       1.46       3.55  2000 1.00
## a_tank[35]  2.58   0.65       1.59       3.50  1625 1.00
## a_tank[36]  1.97   0.48       1.22       2.71  2000 1.00
## a_tank[37]  1.98   0.49       1.23       2.73  2000 1.00
## a_tank[38]  9.99  12.88       1.84      18.57   329 1.00
## a_tank[39]  2.63   0.70       1.48       3.57  1544 1.00
## a_tank[40]  2.24   0.55       1.38       3.09  2000 1.00
## a_tank[41] -2.00   0.52      -2.84      -1.20  2000 1.00
## a_tank[42] -0.58   0.35      -1.10       0.00  2000 1.00
## a_tank[43] -0.44   0.36      -0.99       0.15  2000 1.00
## a_tank[44] -0.32   0.35      -0.94       0.19  2000 1.00
## a_tank[45]  0.64   0.35       0.06       1.17  2000 1.00
## a_tank[46] -0.56   0.34      -1.12      -0.04  2000 1.00
## a_tank[47]  1.98   0.46       1.24       2.70  2000 1.00
## a_tank[48]  0.04   0.32      -0.47       0.54  2000 1.00
## a           1.42   0.30       0.91       1.88  1413 1.00
## sigma       1.03   0.23       0.66       1.37  1325 1.00

plot(precis(m.12h3g, depth = 2))
plot(precis(m.12h3c, depth = 2))
```




```
# alternative version using ggplot2 makes it easy to give both plots the same limits
# using "y" limits because this plot is "flipped" by default
ggplot(precis(m.12h3g, depth = 2)) %>% gf_lims(y = c(-4, 20))
ggplot(precis(m.12h3c, depth = 2)) %>% gf_lims(y = c(-4, 20))
```




```
# this is mainly just to demo the new coeftab behavior
coeftab(m.12h2, m.12h3g, m.12h3c)[c("a", "sigma"), ]

##       m.12h2 m.12h3g m.12h3c
## a       1.30    1.31    1.42
## sigma   1.62    1.63    1.03
```

R Code 6.31

```
library(rethinking)
data(Howell1)
Howell <-
```

```
  Howell1 %>%  mutate(age.s = zscore(age))
set.seed(1000)      # so we all get the same "random" data sets
train <- sample(1:nrow(Howell), size = nrow(Howell) / 2)  # half of the rows
Howell.train <- Howell[ train, ]   # put half in training set
Howell.test  <- Howell[-train, ]   # the other half in test set
```

R Code 6.32

```
# You need to come up with mu and sigma
sum(dnorm(Howell.test$height, mu, sigma, log = TRUE))
```

Additional Problems

1 Making entropy larger.

a) Which is larger: $H(0.1, 0.3, 0.6)$ or $H(0.2, 0.2, 0.6)$?

b) Let $\boldsymbol{p} = \langle p_1, p_2, p_3 \rangle$ and let $\boldsymbol{q} = \langle p, p, p_3 \rangle$ where $p = \frac{p_1+p_2}{2}$. Compute $H(\boldsymbol{p})$ and $H(\boldsymbol{q})$. Which is larger?

c) Suppose a random process has n outcomes. Show that with one exception, there is always another random process that also has n outcomes, but has higher entropy? What is the one exception? (The exception is the random process with the maximal entropy among processes with n outcomes.)

Solution.

```
H <- function(p) - sum(p[p>0] * log(p[p>0]))
H(c(0.1, 0.3, 0.6))

## [1] 0.8979457

H(c(0.2, 0.2, 0.6))

## [1] 0.9502705
```

Let $h(p) = p\log(p) + (s-p)\log(s-p)$ where s is fixed. I've used little h because this is a little part of the full entropy where the probabilities of two events sum to s. I've left off the negative sign to simplify the derivative below.

$h'(p) = \log(p) + p\frac{1}{p} - \log(s-p) - (s-p)\frac{1}{s-p} = \log(p) - \log(s-p)$. So $h'(p) = 0$ when $p = s - p$. This means that the largest entropy happens when these two events have the same probability.

We play this game any time thre are two unequal probabilities, so the the maximal entropy is acheived when all the probabilities are equal.

(This can also be deonstrated by clever algebra and log rules, but I find this more instructive and less messy.)

2 Compute the entropy of tossing two coins two different ways:

a) Consider the outcomes to be 0, 1 or 2 heads.

b) Consider the outcomes to be HH, HT, TH, or TT.

How do the results compare? Can you generalize?