Statistical Rethinking Workbook

Dr. Kim Y. Somfleth

13 July 2022

Abstract

Working through the lectures provided by Richard McElreath, the 2022 version of Statistical Rethinking.

Contents

Introduction

This is my workbook following along the 2022 version of Statistical Rethinking (McElreath 2022), using his R package (McElreath 2020). A lot of the code has been adapted from (Kurz 2021) to use more modern ggplot and tidyverse constructs.

Lecture 1: Drawing the Baysian Owl

Three core goals

1. Understand what you are doing

- So you know every step explicitly, rather than relying on pre-built black boxes (or institutional processes)

2. Document your work to reduce errors

- For future me; for revision and reuse

3. Respectable scientific workflow

- Document, orderly, justifyable reasoning, ie. useful

To draw the baysian owl

1. Theoretical estimand

- What are you trying to do in the first place,
- Not vague metaphorical connection between research, buzzwords, and some datasets.

2. Scientific causal model(s)
   - Generate synthetic observations to be able to probe statistically
3. Build statistical models from (1) and (2)
   - Or whether it is possible at all
4. Simulate (2) to validate (3) yields (1)
   - To justify workflow -> so results are believable
5. Analyse the real data

The reason for using a Bayesian aproach for this is the flexibility, ability to express uncertainty in any situation and direct solutions for measurement errors and missing data. All this without worrying about the procedure and estimator to use.

Science should be before statistics, causal inference is the component that requires the most care. —No causes in, no causes out -Nancy Cartwright—

What is Causal Inference?

```
Causation does not imply correlation
```

- Causal inference is prediction
  - What if I do this? -> causal inference can give the answer
  - Untangle the associations into causes
- Causal inference is *imputation* of missing observations
  - What if I do this again

DAG (Directed Acyclic Graphs)

- Heuristic causal models -> Analysable with your eyeballs (And also just partially ordered **Set**)
- Confound is a variable that affect both sides of a causal relationship
- Used to make transparent scientific assumptions to justify scientific effore expose it to useful critique and connect theories to the Golems (The brainless statistical models)

Lecture 2: Introdution to Bayesian Inference

Figure 1: The famous NASA blue dot from Visible Earth (Visible Earth 2006) which we can think of throwing imaginary asteroids at to sample.

How can we measure what percentage of the earth is covered with water; Spherical uniform sampling (Figure 1). But how do we quantify uncertainty in our measurement? We use Bayesian data analysis.

In essence it is all just counting. The advantage of this is that we can update our most likely conjecture by taking another measurement (Baysian updating).

1. State a causal model for observations arrise, given each possible explanation
2. Count the ways the data could arrive for each explanation

3. Relative plausibility is relative value from (2)

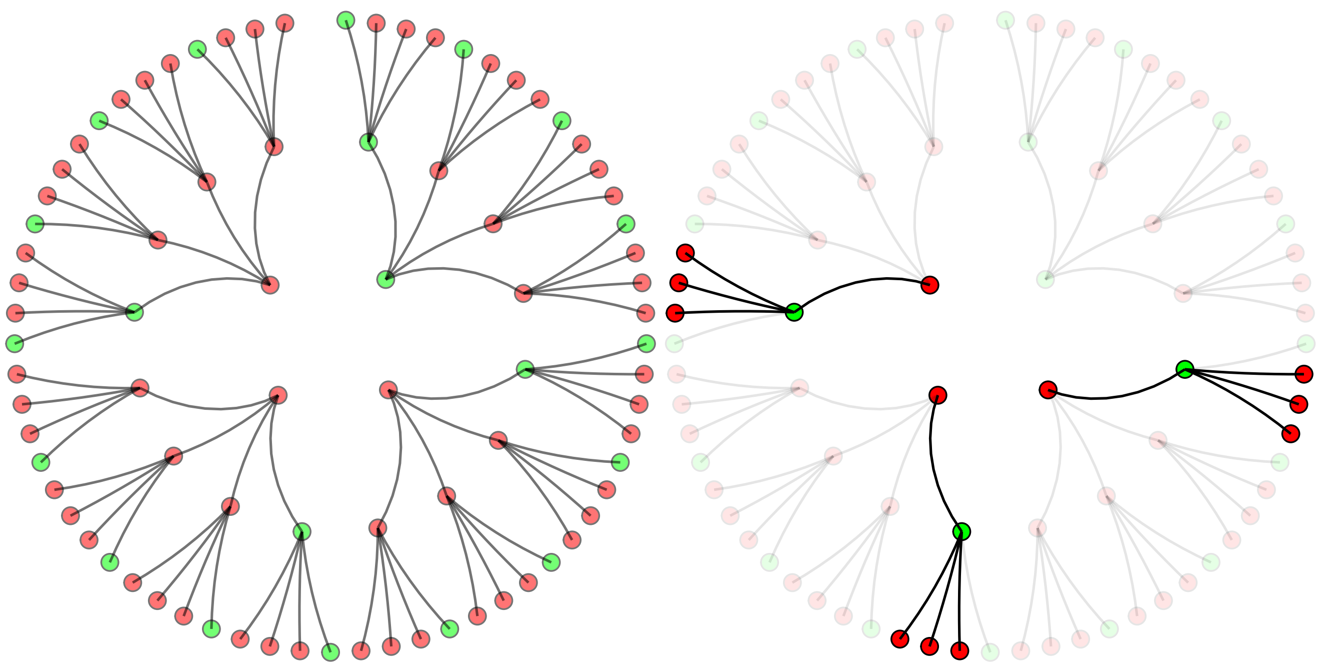
Garden of Forking Data

```
generate_garden <- function(bag,levels,picks=NULL){
    len <- length(bag)
    x   <- seq(0.5,len-0.5,length.out=len)
    if( is.null(picks) ){
        picks = rep(FALSE,levels)
    }

    if( levels == 1 ){
        picked=bag==picks[1]
        return(list(points=tibble(values=bag, x=x, y=1, picked=picked  %>% as.integer()), l
    }else{
        pick      <- picks[1]
        next_pick <- picks[2]
        picked      <- bag == pick
        next_picked <- bag == next_pick
        children_garden <- generate_garden(bag,levels-1,tail(picks,-1))
        furthest_row <- children_garden$points %>% subset(y == max(y))
        closest_row  <- children_garden$points %>% subset(y == min(y))

        x <- x * nrow(furthest_row)
        new_points <- tibble(values=bag, x=x, y=1, picked=picked)
        points_branch <- children_garden$points %>% mutate(y=y+1)
        old_points <- tibble( values = rep(points_branch$values,times=len)
                            , x      = outer(points_branch$x,x,'+') %>% as.vector() - min(x
                            , y      = rep(points_branch$y,times=len)
                            , picked = outer(points_branch$picked,picked,'*') %>% as.vecto
        nearest_points <- old_points %>% subset(y == min(y))
        points <- bind_rows(new_points,old_points)

        if( levels > 2 ){
            lines_branch <- children_garden$lines %>% mutate(y_start=y_start+1,y_end=y_end+
            old_points_in_branch <- old_points %>% subset(y > min(y))
            old_lines <- tibble( y_start = old_points_in_branch$y - 1
                               , y_end   = old_points_in_branch$y
                               , x_start = outer(lines_branch$x_start,x,'+')      %>% as.ve
                               , x_end   = outer(lines_branch$x_end  ,x,'+')      %>% as.ve
                               , picked  = outer(lines_branch$picked ,picked,'*') %>% as.ve
        }else{
            old_lines <- tibble()
        }
        new_lines <- tibble( y_start=1, y_end=2
```

```
                            , x_start=rep(x,times=rep(len,len))
                            , x_end  =nearest_points$x
                            , picked =outer(next_picked,picked,'*') %>% as.vector() %>% as.i
        lines <- bind_rows(new_lines,old_lines)
        return(list(points=points,lines=lines))
    }
}

combine_gardens <- function(gardens,sep=2){
    points <- gardens %>% lapply(function(x) x$points) %>% bind_rows(.id='chunk')
    lines  <- gardens %>% lapply(function(x) x$lines ) %>% bind_rows(.id='chunk')
    gaps <- points %>%
        group_by(chunk) %>%
        subset(y==max(y)) %>%
        summarise(n=n()) %>%
        mutate(n=lag(n+sep,n=1,default=sep/2)) %>% #c(sep/2,n+sep) %>% head(-1)) %>%
        mutate(ac=cumsum(n)) %>%
        select(-n)
    points <- left_join(points,gaps,by='chunk') %>% mutate(x=x+ac)
    lines  <- left_join(lines ,gaps,by='chunk') %>% mutate(x_start=x_start+ac,x_end=x_end+a
    vlines <- gaps %>% mutate(ac = ac - sep/2)  %>% pull(ac)
    return(list(points=points,lines=lines,vlines=vlines))
}

draw_garden <- function(points,lines,pick_alpha=TRUE,vertical_lines=NULL){
    colours <- points$values %>% unique()
    values2colour <- ggthemes::tableau_color_pal()(colours %>% length()) %>% setNames(colou
    leaves <- points %>% subset(y == max(y))
    levels <- points %>% pull(y) %>% max()
    #garden$points <- garden$points %>% mutate(fill=values2colour[values] %>% as.vector())

    if( !pick_alpha ){
        points$picked <- 1
        lines$picked  <- 1
    }

    p <- points %>%
        ggplot(aes(x=x,y=y)) +
        geom_point(aes(fill=values,alpha=picked),shape=21, size=3) +
        geom_segment(aes(x=x_start,xend=x_end,y=y_start,yend=y_end,alpha=picked),data=lines
        coord_polar() +
        scale_x_continuous(limits=c(0,leaves$x %>% max %>% ceiling + 1) ,breaks=NULL) +
        scale_y_continuous(limits=c(1/levels,levels+1),breaks=NULL) +
        scale_fill_manual(values=colours) +
        theme( legend.position = "none"
```

```
             , panel.grid = element_blank()
             , axis.title = element_blank()
             , panel.background = element_rect(fill = "transparent",colour = NA)
             , plot.background = element_rect(fill = "transparent",colour = NA) )
    if( !is.null(vertical_lines) ){
        p <- p + geom_vline(xintercept = vertical_lines, colour='black' )
    }
    return(p)
}
```

We start in the simpler case of finite possibilities. Suppose we are picking (with replacement) marbles out of a bag, if we have four marbles; one green, three red, we can determine all possibilities of getting three marbles (Figure 2).



(a) Universe of possible picks (b) Universe after picks

Figure 2: The branching universe of possibilities of picking 3 marbles with replacement in green, red, red, red, as well as the possibilities for picking red, green, red.

Now instead suppose we wanted to find out the proportion of marbles in the bag without prior knowledge, instead we examine the relative likelihood of different possibilities of combinations within the bag (Figure 3).

Posterior: Bayes' Theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

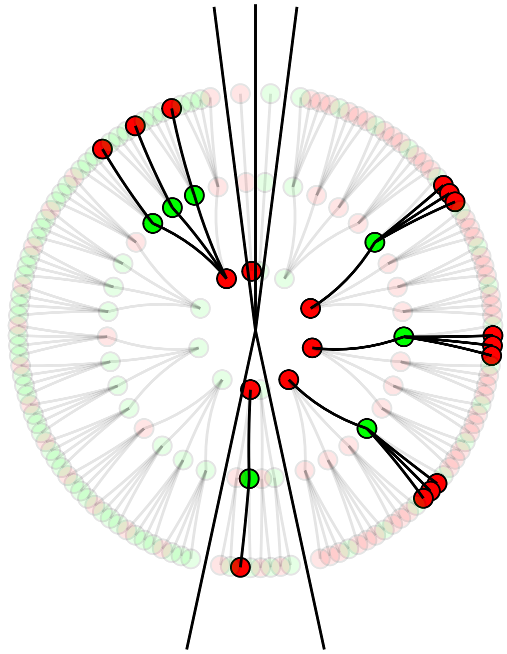


Figure 3: The universes of possible two colour four marble equivalent bags.

Can think of it as updating our prior knowledge of the answer $P(A)$ with new measurement $P(B|A)$, giving the new prior $P(A|B)$ given the new data B.

For instance in our water case the distribution based on uniform samples is just

$$(1-x)^n x^m$$

for n instances of land and m instances of water. This distribution is a binomial distribution (Figure 4).

$$P(W,L|p) = \frac{(W+L)!}{W!L!} p^W (1-p)^L$$

or

$$P(WinN|p) = \frac{N!}{W!(N-W)!} p^W (1-p)^{N-W}.$$

So Bayes' gives

$$P(p|W,L) = \frac{P(W,L|p)P(p)}{P(W,L)}$$

making the updating step much easier to understand. Note that a flat prior, ie each p between 0 and 1 is equally likely; a flat prior.

So the next question how does one report a result from such a distribution. Can one use the mean or median; well generally no the distribution is the answer, and these —point estimates— remove some of the complexity of your data, of course publications may want you to present such an arbitrary value. Well then maybe a confidence interval to communicate some of the shape; again no, but somewhat more useful, it is just a reduction of the distribution. For instance a 50% interval doesn't really describe the data that well as the center or first or last 50% are equally valid. Something like a 99% interval does have more of a use describing how well something is described, but it is really just arbitrary – **The distribution is the answer** – 95% intervals don't have anything to do with robustness.

```
ps = seq(0,1,length.out=1000)
prior = rep_along(ps,1) #flat prior
labels = list('1'='1/1','2'='3/6','3'='5/8')
d = data.frame( ps=ps
              , ys1 = dbinom(1,1,ps)*prior
              , ys2 = dbinom(3,6,ps)*prior
              , ys3 = dbinom(5,8,ps)*prior ) %>%
    mutate( ys1 = ys1/sum(ys1), ys2=ys2/sum(ys2), ys3=ys3/sum(ys3) ) %>%
    pivot_longer( ys1:ys3, names_to='dist', names_prefix='ys', values_to='ys')
ggplot(d) +
    geom_line(aes(x=ps,y=ys,colour=dist), size=2) +
    scale_colour_manual(name='Samples Water', labels=c('1/1','3/6','5/8'), values=ggthemes
```

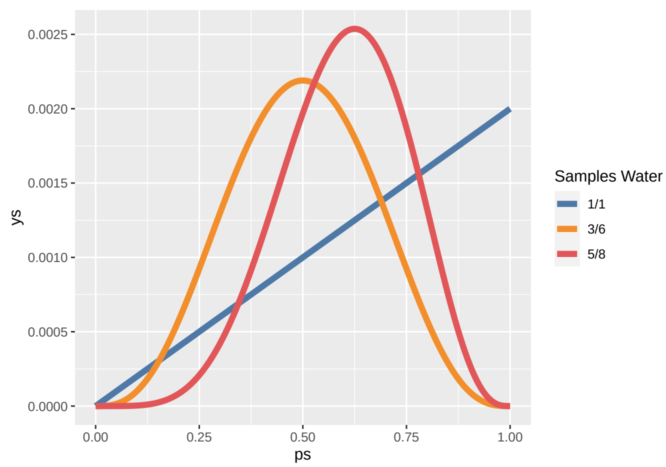


Figure 4: Binomial distributions for proportion of true given specified true and total measurements.

From Posterior to Prediction

To make actual predictions, we model of the posterior distribution against some question. The simple way is to take samples of the posterior and create a predictive distribution given the sampled p for a number of discrete samples n. This predictive distribution is then sampled to construct the posterior predictive distribution through accumulation. These two steps are just rbinom in R, see (Figure 5), completed also for the prior, showing the results expected by a repeat experiment.

```
n_samples=1e4
w = 6
size = 9

ps = seq(0,1,length.out=n_samples)
prior = rep_along(ps,1)
probability = dbinom( w, size=size, prob=ps )
```

```r
posterior = probability * prior
posterior = posterior / sum(posterior)

samples = sample( ps, prob=posterior, size=n_samples, replace=TRUE )
posterior_predictive = rbinom( n_samples, size=size, prob=samples )

samples_prior = sample( ps, prob=prior/sum(prior), size=n_samples, replace=TRUE )
prior_predictive = rbinom( n_samples, size=size, prob=samples_prior )

d = data.frame( x   =c(posterior_predictive,prior_predictive)
              , type=rep( c('Posterior Predictive','Prior Predictive')
                        , times=c(n_samples,n_samples)))

d %>%
    ggplot(aes(x=x,colour=type,fill=type)) +
    geom_histogram(binwidth=1,position='dodge') +
    labs(title=paste0(w,' wins in ',size,' samples posterior predictive'))
```

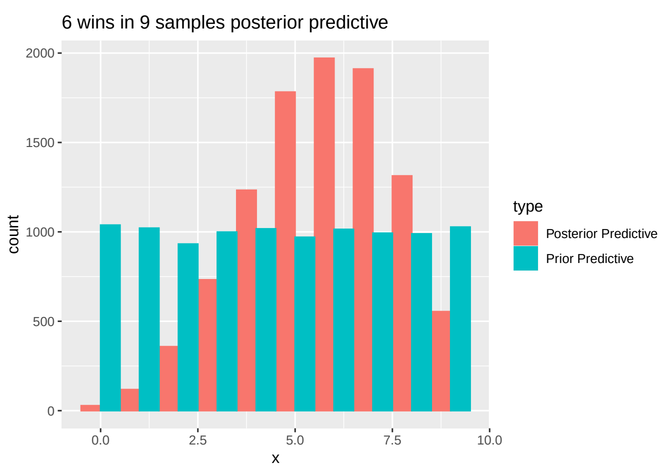


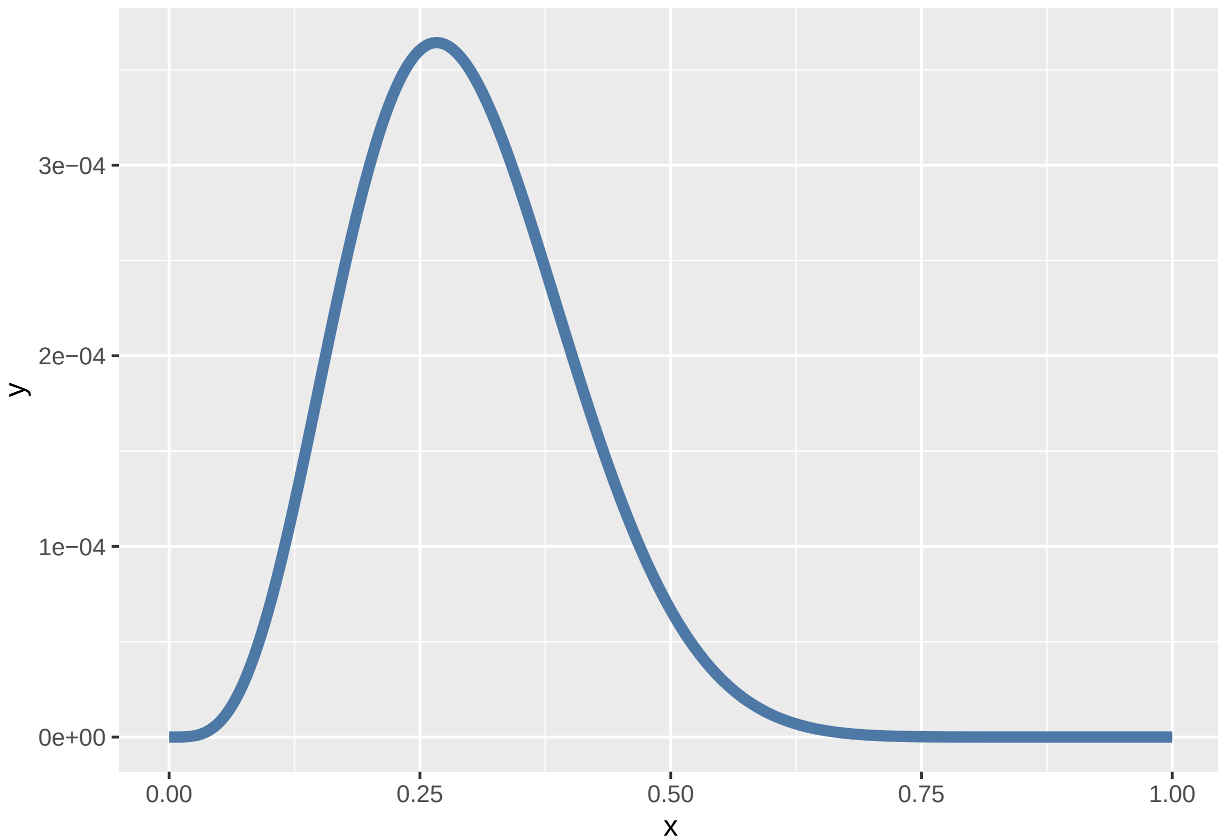
Figure 5: (ref:PosteriorPredictiveFigureCaption)

Week 1 Homework

1. Suppose the globe toss data had been 4 water, 11 land. Construct the posterior distribution (using grid approximation) with flat prior.

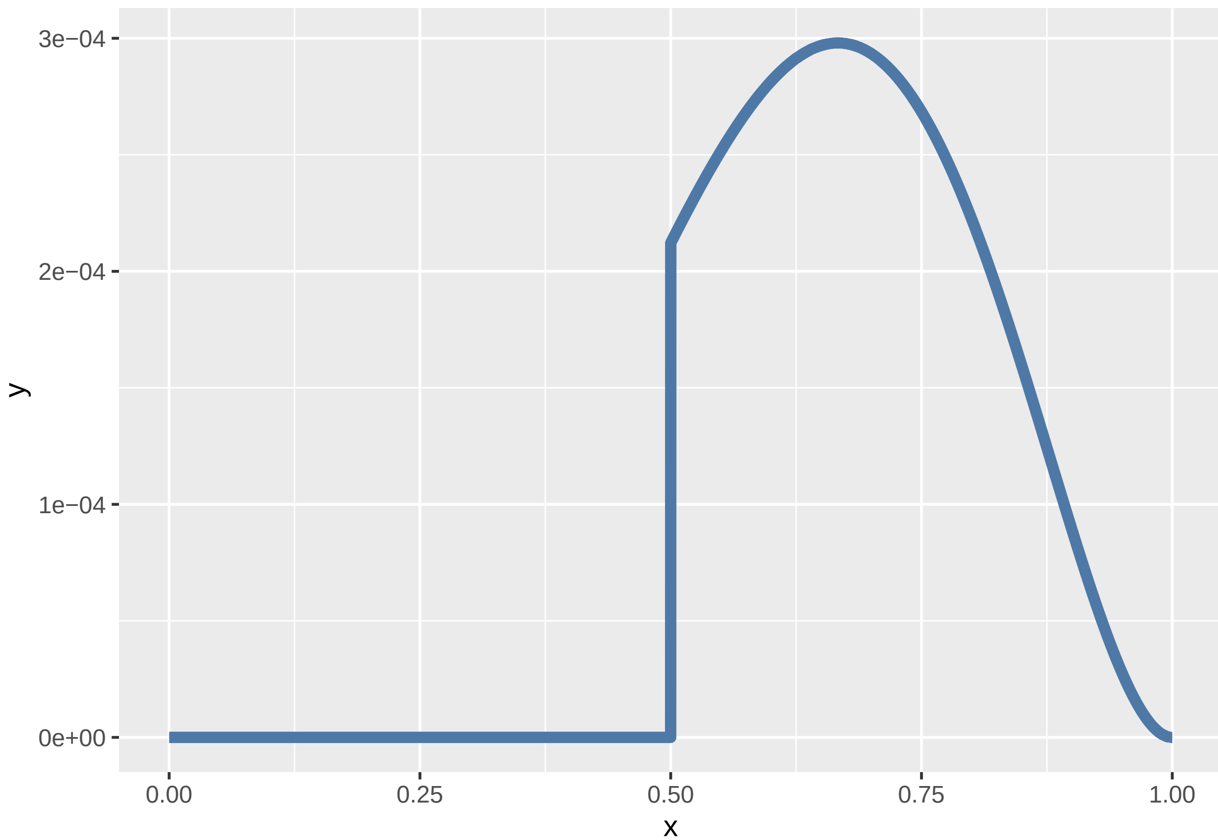
```r
water = 4
land = 11

n_samples = 1e4

ps = seq(0,1,length.out=n_samples)
prior = rep_along(ps,1)
```

```
probability = dbinom(water,water+land,ps)
posterior = probability * prior
posterior = posterior / sum(posterior)

d = data.frame( x=ps, y=posterior )
ggplot(d) +
    geom_line(aes(x=x,y=y), colour=ggthemes::tableau_color_pal()(1), size=2)
```



2. The same but 4 water, 2 land and step prior at $p = 0.5$

```
water = 4
land = 2

n_samples = 1e4

ps = seq(0,1,length.out=n_samples)
prior = rep(c(0,1),times=c(n_samples/2,n_samples/2))
probability = dbinom(water,water+land,ps)
posterior = probability * prior
posterior = posterior / sum(posterior)

d = data.frame( x=ps, y=posterior )
ggplot(d) +
    geom_line(aes(x=x,y=y), colour=ggthemes::tableau_color_pal()(1), size=2)
```



3. Compute (assumed central) 89% percentile and HPDI intervals from (2).

```
samples = sample( ps, prob=posterior, size=n_samples, replace=TRUE )
c(HPDI(samples),PI(samples))
```

```
##      |0.89     0.89|        5%       94%
## 0.5000500 0.8434843 0.5254525 0.8809881
```

Lecture 3: Geocentric Models

Geocentric models are using multiple orbits to explain the orbits of the planets, essentially just a fourier technique. This is very much the same as linear regression. Both are suprisingly accurate portrayals of observation with no mechanistic justification.

The Baysian argument for prediction

Ancient argument by Gaus; Gaus' distribution

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{x^2}{\sigma^2}}.$$

This allows us to define a mean and standard deviation, even on data that isn't really normal. So mean and standard deviation can be defined, but not generatively related. Like Feynman said, the name of a bird is kinda useless on it's own, but it is useful to communicate about it to other people.

Syntax for Modelling

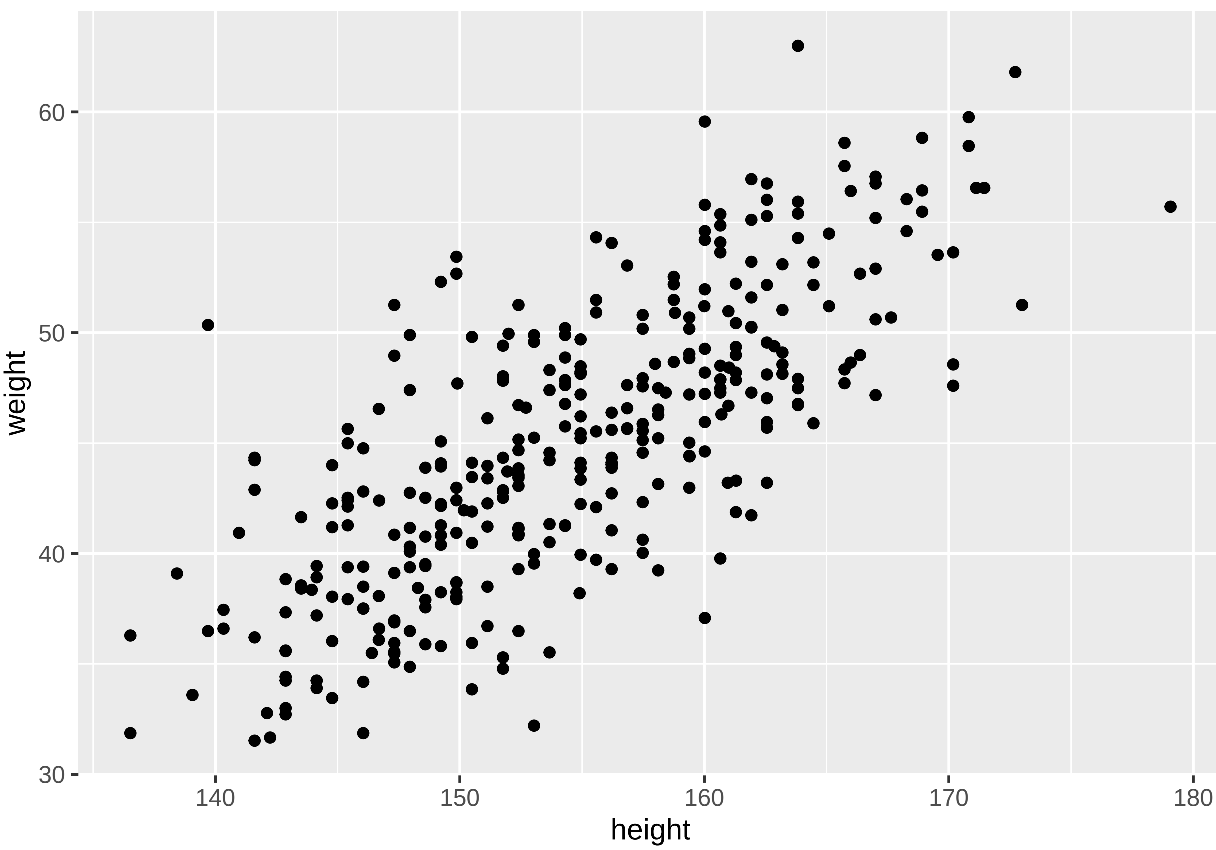
$$W \sim Binomial(N, p)$$
$$p \sim Uniform(0, 1)$$

W is the outcome, is distributed, $Binomial(N, p)$ is the distribution, and p the prior. In conditional expression

$$Pr(W|N, p) = Binomial(W|N, p)$$
$$Pr(p) = Uniform(p|0, 1)$$

Linear Generative Models

```r
data(Howell1)
d <- Howell1 %>% subset(age > 18)
d %>%
    ggplot(aes(x=height,y=weight)) +
    geom_point()
```



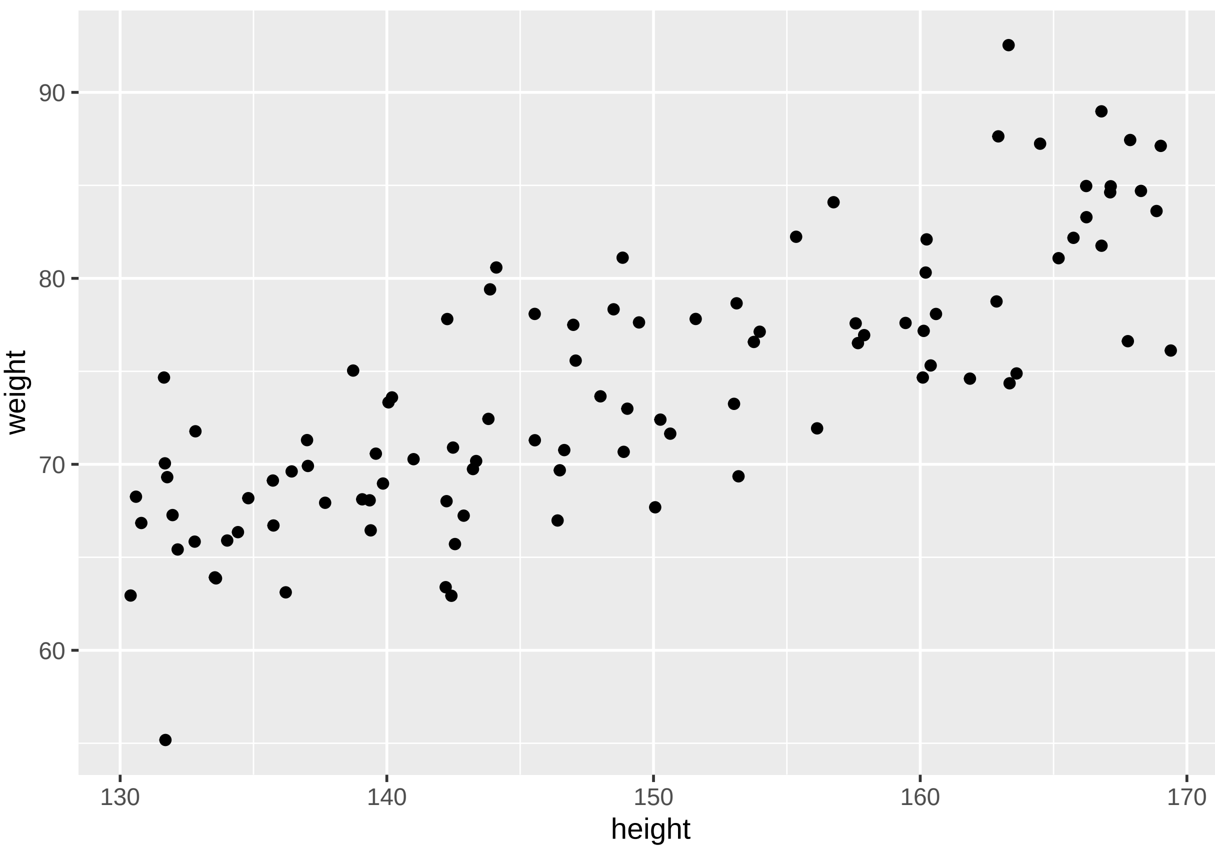
As an example we will lok at height vs weight, with weigth being depednent on height, but not the other way around. The linear model is

$$y_i \sim Normal(\mu_i, \sigma)$$

$$\mu_i = \alpha + \beta x_i$$

Generative Model H -> W

```
alpha <- 0
beta  <- 0.5 #kg/cm
sigma <- 5
n_individuals <- 100

H  <- runif(n_individuals,130,170) #cm
mu <- alpha + beta * H
W  <- rnorm(n_individuals,mu,sigma)

d_gen <- data.frame(height=H,weight=W)
d_gen %>%
    ggplot(aes(x=height,y=weight)) +
    geom_point()
```



Statistical Linear Model

To fit we need priors

$$\alpha \sim Normal(0, 1)$$
$$\beta \sim Normal(0, 1)$$

$$\sigma \sim Normal(0, 1)$$

1. It is useful to rescale variables
   - Makes the simulation and priors easier as well as integration downstream
   - $H_i -> H_i - \bar{H}$: α becomes average weight.
2. Must think about priors

Using this we set new priors

$$\alpha \sim Normal(60, 10)$$

$$\beta \sim Normal(0, 10)$$

$$\sigma \sim Normal(0, 10)$$

which all have huge variance, ie we want to learn these variables. In fact if we sample our priors we get really weird relationships between height and weight, we can switch β to

$$\beta \sim LogNormal(0, 1)$$

which both always positive and favours smaller slopes, a basic biological constraint. We still want to have the prior cover all plausible possible fits for the type of data we are looking at and to only be constraind from outside data.

For linear models, the prior doesn't really matter after a quite small sample set, however this is for practice in more complicated settings. In fact we will start determining priors from the data we are fitting!

There is one small problem generating posterior distributions

$$P(\alpha, \beta, \sigma | W, H) \propto Normal(W|\mu, \sigma) Normal(\alpha|60, 10) LogNormal(\beta|0, 1) Normal(\sigma|0, 10)$$

it blows up in number of samples to uniformly sample it.

Gaussian Approximation

Posterior distributions are approximately Gaussian $\rightarrow$ Use Gaussian approximation (often called quadratic or laplace approximation)

Validation

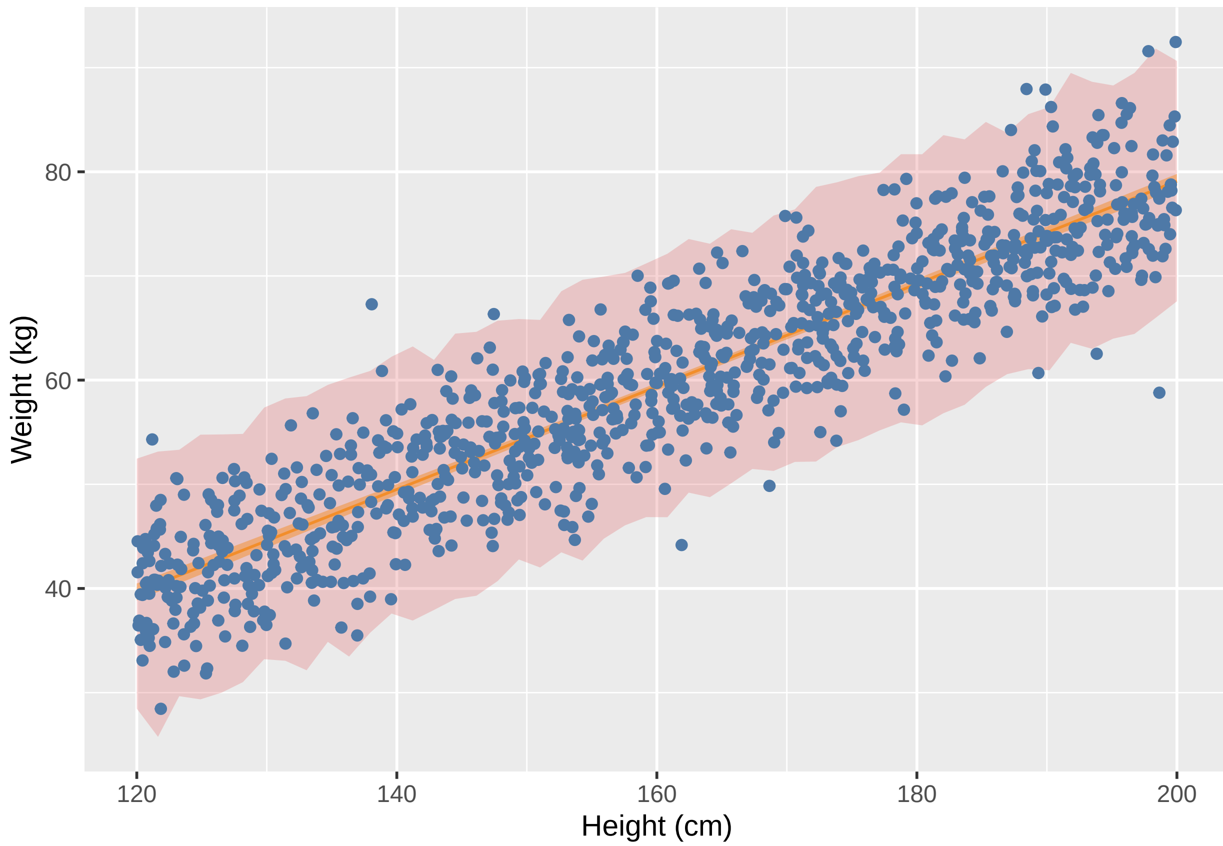
At a minimum take your simulated data, fit it using your methodology and check the output against your synthetic input data. Then run it on your data.

First law of Statistics

Resultant parameters are not independant! Instead push out posterior predictions instead and describe/interpret those.

```r
alpha     <- 60
beta      <- 0.5
sigma     <- 5
n_samples <- 1000

h_min <- 120
h_max <- 200

H  <- runif(n_samples,h_min,h_max)
mu <- alpha + beta*(H-mean(H))
W  <- rnorm(n_samples,mu,sigma)

d_sim <- data.frame(height=H,weight=W)
l_sim <- list(H=H, W=W, Hbar=mean(H))

fit <- quap( alist( W ~ dnorm(mu,sigma)
                  , mu <- a + b * (H-Hbar)
                  , a     ~ dnorm(70,10)
                  , b     ~ dlnorm(0,1)
                  , sigma ~ dunif(0,10) )
           , data=l_sim )



hs <- seq(h_min,h_max,length.out=50)
fit_data <- list(H=hs,Hbar=mean(H))
mu <- link(fit,data=fit_data)
mu_mean <- colMeans(mu)
mu_ci <- apply(mu,2,quantile,probs=c(0.005,0.995))

d_fit = data.frame(height=hs,weight_mean=mu_mean,weight_lower=mu_ci[1,],weight_upper=mu_ci

W_sim  <- sim(fit,data=fit_data)
W_mean <- colMeans(W_sim)
W_ci   <- apply(W_sim,2,quantile,probs=c(0.005,0.995))

d_sime = data.frame(height=hs,weight_mean=W_mean,weight_lower=W_ci[1,],weight_upper=W_ci[2

p <- d_sim %>%
    ggplot(aes(x=height,y=weight)) +
    geom_ribbon(aes(x=height,y=weight_mean,ymin=weight_lower,ymax=weight_upper), fill=ggthe
    geom_ribbon(aes(x=height,y=weight_mean,ymin=weight_lower,ymax=weight_upper), fill=ggthe
    geom_line(aes(x=height,y=weight_mean),colour=ggthemes::tableau_color_pal()(2)[2],data=d
    geom_point(colour=ggthemes::tableau_color_pal()(1)) +
    xlab('Height (cm)') + ylab('Weight (kg)') +
```

```
    xlim(h_min,h_max)
p
```



Lecture 4: Categorical variables and curve/spline fitting

Categorical Variables

There are two equivalent ways of defining categorical variables; through dummy variables or through index variables. The latter is more easily implemented and also applicable to multi level models. Another advantage of index variables is ease of inclusion of additional categories. They work by assigning a number to each category. For instance:

$$W \sim Normal(\mu_i, \sigma)$$

$$\mu_i = \alpha_{S[i]}$$

for $S = [\alpha_1, \alpha_2]$.

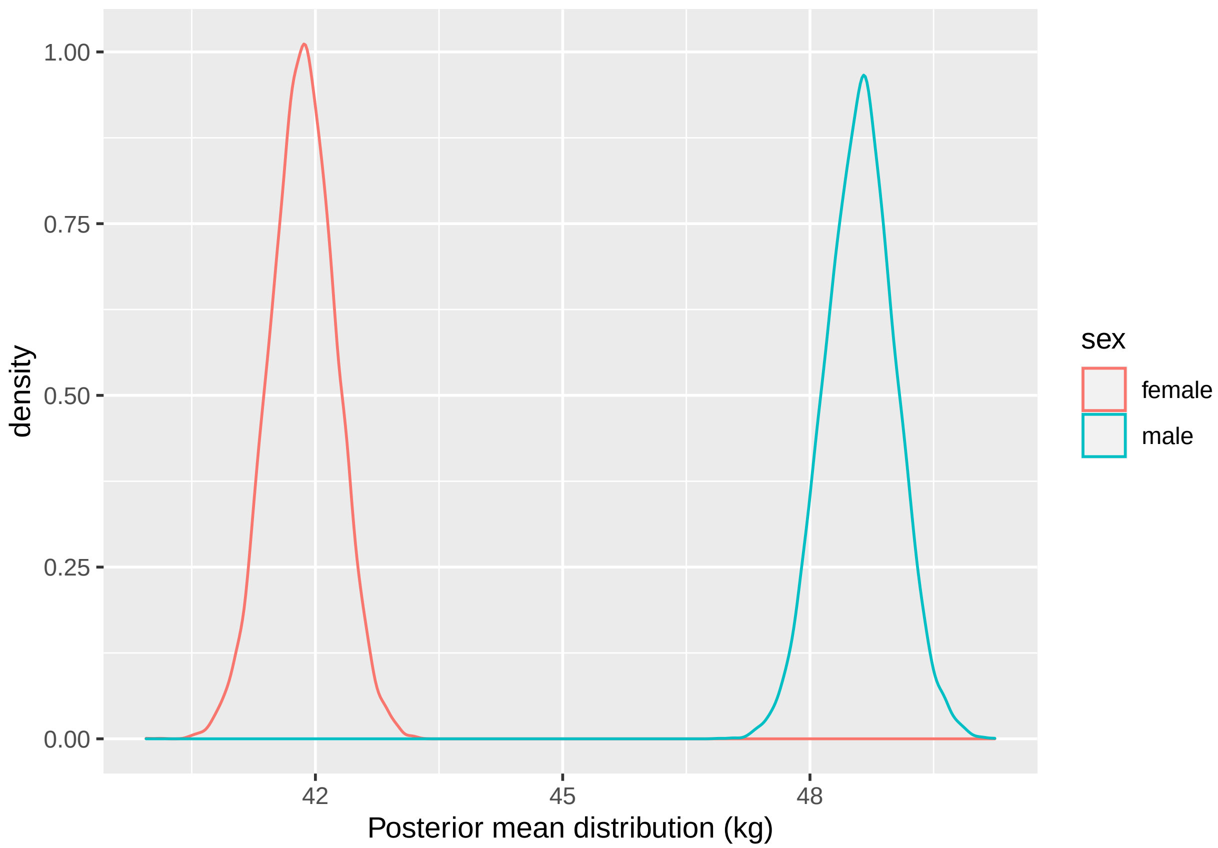
We can apply this to the height data incorporating sex.

```
data("Howell1")
d = Howell1 %>% subset(age >= 18)

dat = list(W=d$weight, S=d$male+1)
```

```
cat_map = c('female','male')

m_SW = quap(
    alist(
        W ~ dnorm(mu,sigma),
        mu <- a[S],
        a[S] ~ dnorm(60,10),
        sigma ~ dunif(0,10)
    ),
    data=dat
)
```

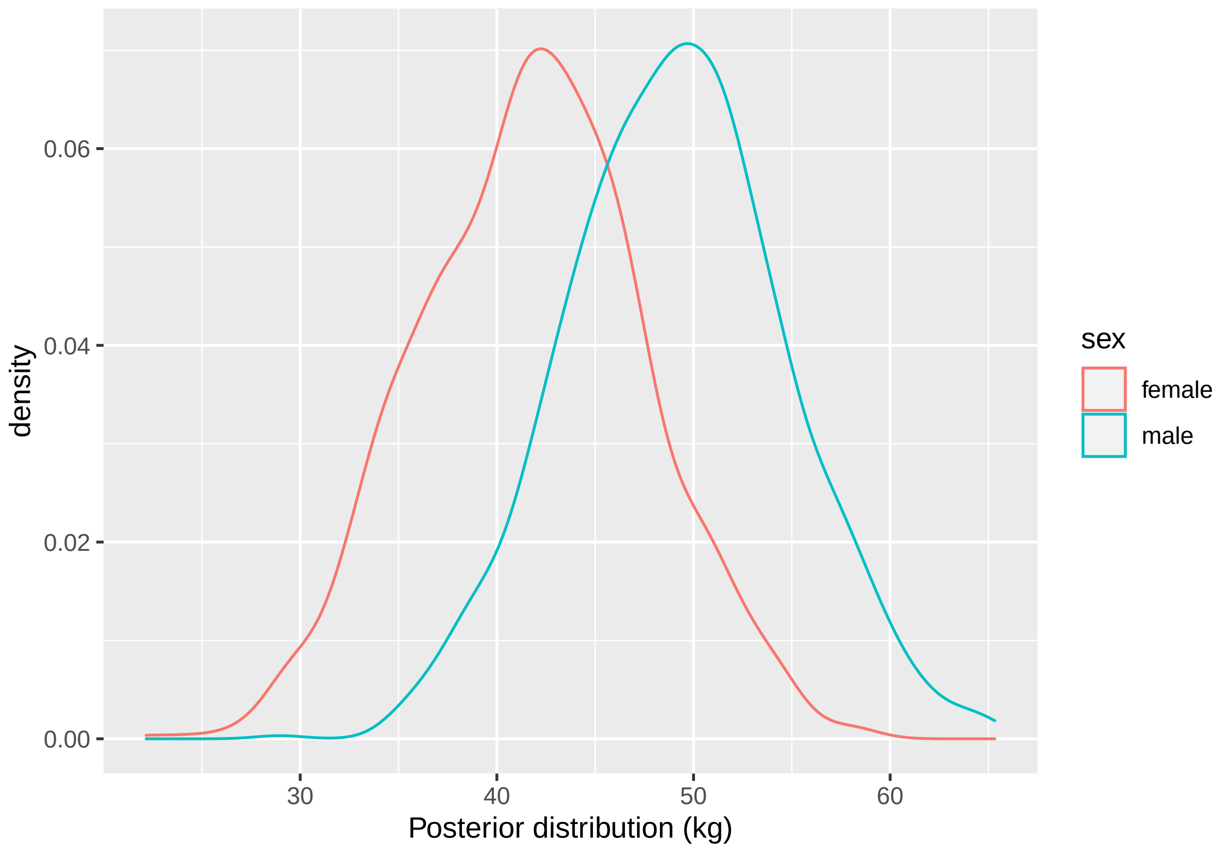
The posterior predictions can be constructed in the same way as before.

```
post = extract.samples(m_SW) %>%
    as.data.frame()
post_longer = post %>%
    pivot_longer(-sigma,names_to='sex',names_prefix='a.',values_to='weight') %>%
    mutate(sex=factor(sex,levels=c(1,2),labels=c('female','male')))

post_longer %>% ggplot(aes(x=weight,colour=sex)) +
    geom_density() +
    xlab('Posterior mean distribution (kg)')
```



However there is a distinction, we also have difference in distribution.

```
n_samples = 1000
post2 = data.frame(W.1=rnorm(n_samples,post$a.1,post$sigma)
                   ,W.2=rnorm(n_samples,post$a.2,post$sigma))
post2_longer = post2 %>%
    pivot_longer(everything(),names_to='sex',names_prefix='W.',values_to='weight') %>%
    mutate(sex=factor(sex,levels=c(1,2),labels=c('female','male')))

post2_longer %>% ggplot(aes(x=weight,colour=sex)) +
    geom_density() +
    xlab('Posterior distribution (kg)')
```

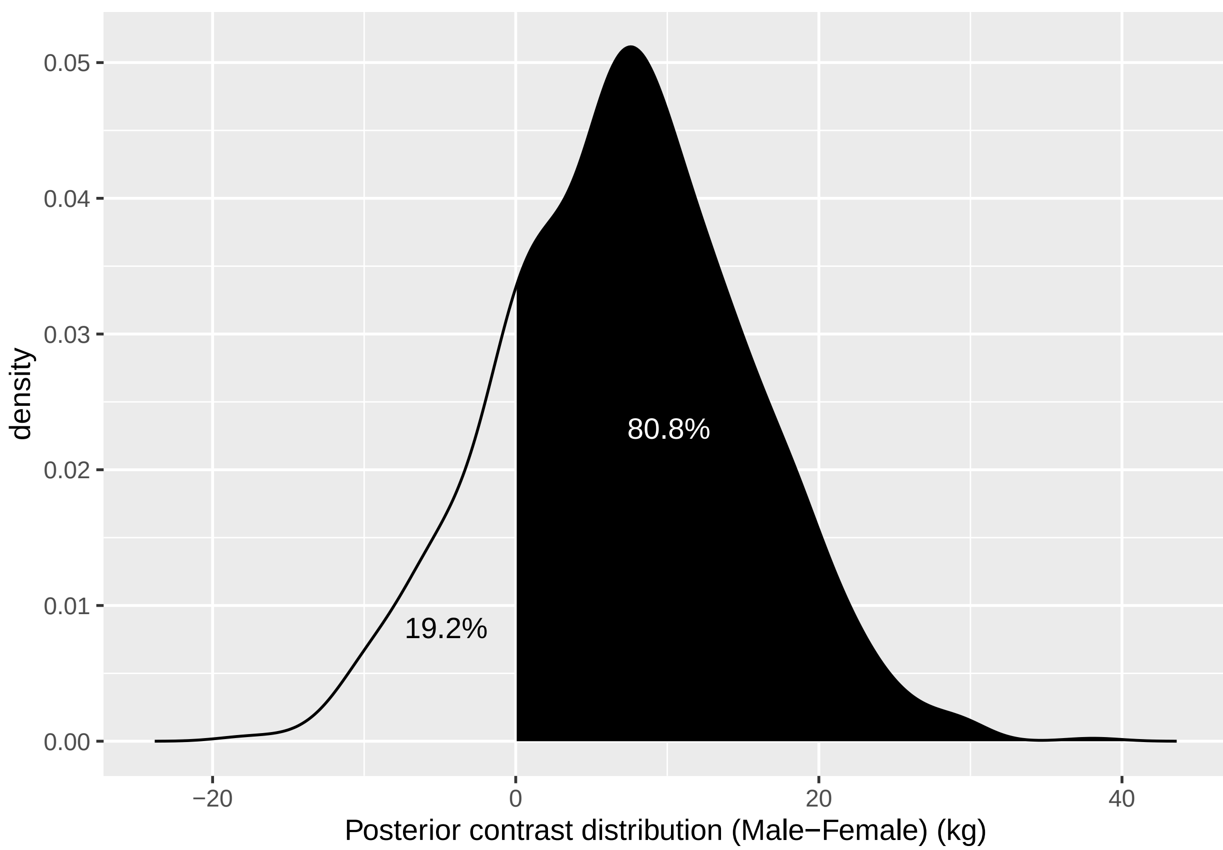


Which one must compare using contrast.

```
post_contrast = (post2$W.2 - post2$W.1) %>%
    density() %>%
    (function(x) data_frame(dW=x$x,density=x$y)) %>%
    mutate(colour=if_else(dW>0,'white','black'))
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```r
labels = post_contrast %>%
    group_by(colour) %>%
    summarise(x=weighted.mean(dW,density),label=sum(density)) %>%
    mutate(y=approx(post_contrast$dW,post_contrast$density,x)$y/2
          ,label=paste0(round(label/sum(label)*100,digits=1),'%'))

post_contrast %>%
    ggplot(aes(x=dW,y=density)) +
    geom_line() +
    geom_area(data=filter(post_contrast,dW>0),fill='black') +
    geom_text(aes(x=x,y=y,colour=colour,label=label),data=labels) +
    scale_colour_manual(values=labels$colour) +
    xlab('Posterior contrast distribution (Male-Female) (kg)') +
    theme(legend.position="none")
```



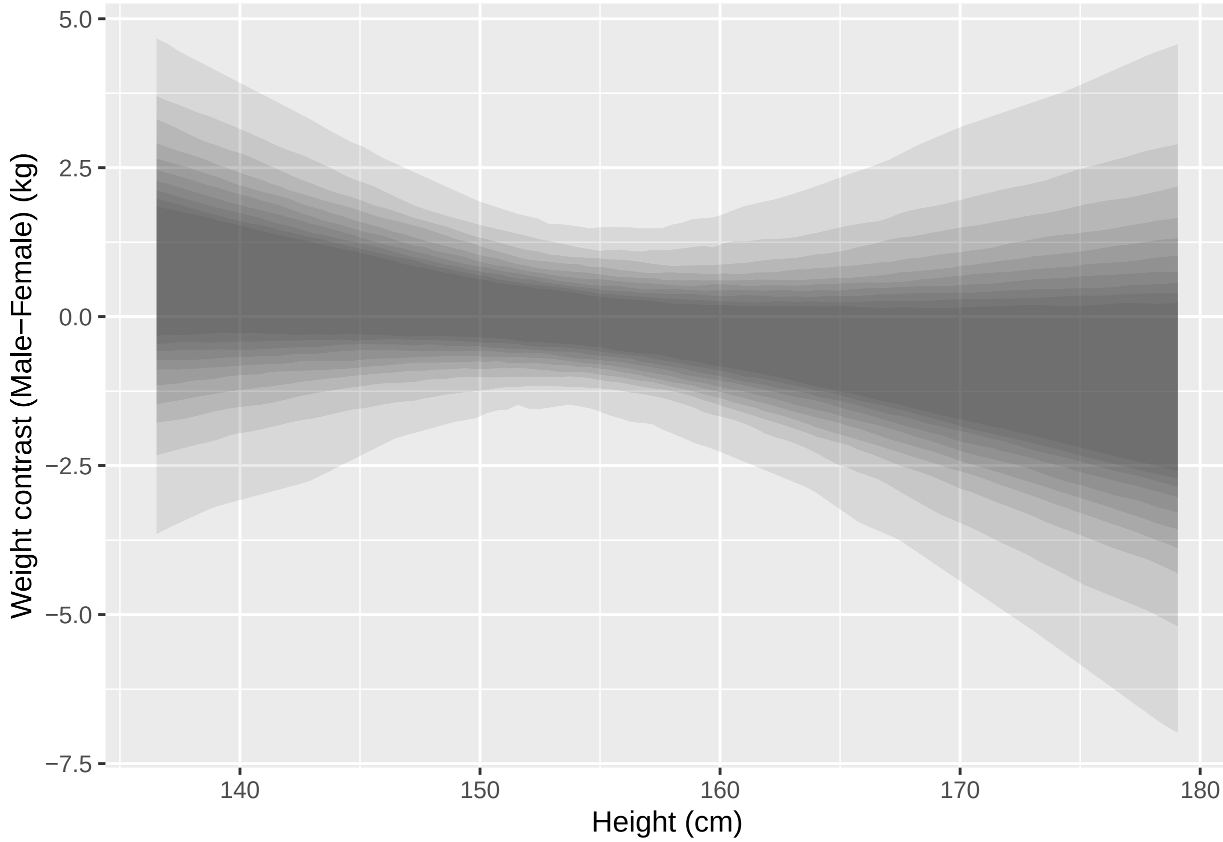
Adding Regression

```r
data("Howell1")
d = Howell1 %>% subset(age >= 18)

datr = list(W=d$weight, H=d$height,  Hbar=mean(d$height), S=d$male+1)
cat_map = c('female','male')

m_SHW = quap(
```

```
    alist(
        W ~ dnorm(mu,sigma),
        mu <- a[S] + b[S]*(H-Hbar),
        a[S] ~ dnorm(60,10),
        b[S] ~ dlnorm(0,1),
        sigma ~ dunif(0,10)
    ),
    data=datr
)
```

The contrast for weight to height would then look like

```
hs = seq(from=min(d$height),to=max(d$height),len=100)

w_f = link(fit=m_SHW,data=list(S=rep_along(hs,1),H=hs,Hbar=mean(d$height)))
w_m = link(fit=m_SHW,data=list(S=rep_along(hs,2),H=hs,Hbar=mean(d$height)))
w_c = w_m - w_f

intervals = seq(from=0.5,to=0.99,len=10) %>%
    map_dfr(~apply(w_c,2,PI,prob=.x) %>%
                t %>%
                as_tibble() %>%
                set_names(c('ymin','ymax')) %>%
                mutate(hs=hs),.id='i')

ggplot(intervals,aes(x=hs,ymin=ymin,ymax=ymax,group=i)) +
    geom_ribbon(alpha=0.1) +
    xlab('Height (cm)') +
    ylab('Weight contrast (Male-Female) (kg)')
```



The full Bayes

```
data("Howell1")
d = Howell1 %>% subset(age >= 18)

datf = list(W=d$weight, H=d$height,  Hbar=mean(d$height), S=d$male+1)
cat_map = c('female','male')

m_SHW_full = quap(
    alist(
        W ~ dnorm(mu,sigma),
        mu <- a[S] + b[S]*(H-Hbar),
        a[S] ~ dnorm(60,10),
        b[S] ~ dlnorm(0,1),
        sigma ~ dunif(0,10),

        H ~ dnorm(nu,tau),
        nu <- h[S],
        h[S] ~ dnorm(160,10),
        tau ~ dunif(0,10)
    ),
    data=datf
```

```
)
precis(m_SHW_full,depth=2)
```

```
##                mean         sd        5.5%       94.5%
## a[1]    45.1671915 0.43697408  44.4688225  45.8655604
## a[2]    45.0947070 0.45575014  44.3663303  45.8230838
## b[1]     0.6567818 0.06083098   0.5595622   0.7540015
## b[2]     0.6096394 0.05480441   0.5220514   0.6972274
## sigma    4.2279407 0.15934876   3.9732706   4.4826108
## h[1]   149.5307135 0.40342800 148.8859577 150.1754694
## h[2]   160.3589377 0.42943310 159.6726207 161.0452548
## tau      5.5212613 0.20808719   5.1886978   5.8538248
```

```
#HW_sim = sim(m_SHW_full,data=list(S=c(1,2),Hbar=datf$Hbar),vars=c('H','W'))
```

Week 2 Homework

1. Construct a linear regression of weight as predicted by height, using the adults (age 18 or greater) from the Howell1 dataset. The heights listed below were recorded in the !Kung census, but weights were not recorded for these individuals. Provide predicted weights and 89% compatibility intervals for each of these individuals. That is, fill in the table below, using model-based predictions.

```
n_samples = 1e4
q1 = tibble(height=c(140,160,175))

data("Howell1")
d = Howell1 %>% subset(age >= 18)

datq1 = list(W=d$weight, H=d$height,  Hbar=mean(d$height))

m_HW = quap(
    alist(
        W ~ dnorm(mu,sigma),
        mu <- a + b*(H-Hbar),
        a ~ dnorm(60,10),
        b ~ dlnorm(0,1),
        sigma ~ dunif(0,10)
    ),
    data=datq1
)

q1_sim = sim(m_HW,data=list(H=q1$height,Hbar=datf$Hbar),vars=c('W'))
q1$pw = apply(q1_sim,2,mean)
q1 = cbind(q1,t(apply(q1_sim,2,PI)))
```

```r
knitr::kable(q1)
```

height	pw	5%	94%
140	35.88266	28.94987	42.19580
160	48.17965	41.32301	54.99757
175	57.66095	51.10271	64.68449

2. From the Howell1 dataset,consider only the people younger than 13 years old. Estimate the causal association between age and weight. Assume that age influences weight through two paths. First, age influences height, and height influences weight. Second, age directly influences weight through age- related changes in muscle growth and body proportions.

$$
\begin{aligned}
W &\sim Normal(\mu, \sigma) \\
\mu &= \alpha + \beta A \\
\alpha &\sim Normal(60, 10) \\
\beta &\sim LogNormal(0, 1) \\
\sigma &\sim Uniform(0, 10)
\end{aligned}
$$

```r
data("Howell1")
d = Howell1 %>% subset(age < 13)

datq2 = list(W=d$weight, A=d$age)
cat_map = c('female','male')

m_q2 = quap(
    alist(
        W ~ dnorm(mu,sigma),
        mu <- a + b*A,
        a ~ dnorm(4,2),
        b ~ dlnorm(0,1),
        sigma ~ dunif(0,10)
    ),
    data=datq2
)
precis(m_q2)
```

```
##           mean         sd     5.5%    94.5%
## a     7.351637 0.35681483 6.781377 7.921896
## b     1.352455 0.05426189 1.265734 1.439176
## sigma 2.524734 0.14781645 2.288495 2.760974
```

3. Effect of sex on weight

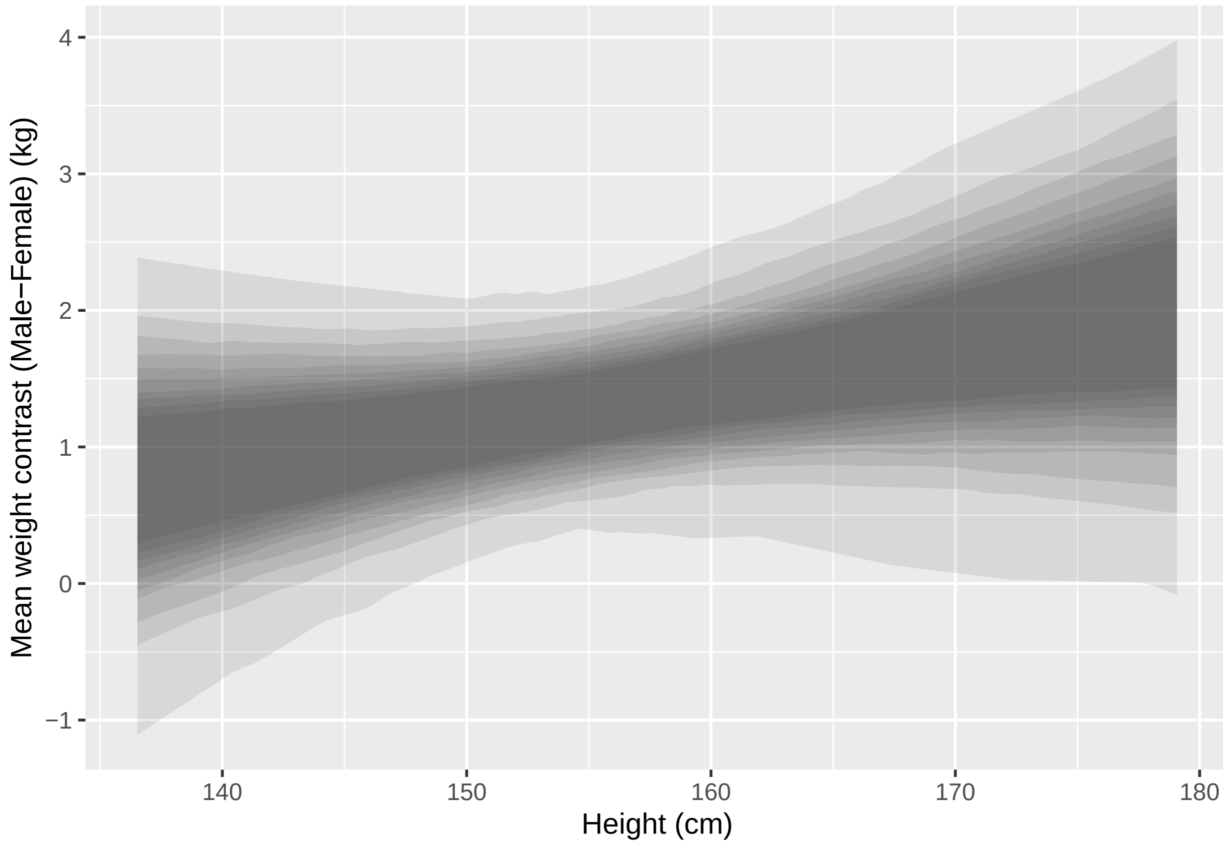
$$W \sim Normal(\mu, \sigma)$$
$$\mu = \alpha_{S[i]} + \beta_{S[i]} A$$
$$\alpha_{S[i]} \sim Normal(60, 10)$$
$$\beta_{S[i]} \sim LogNormal(0, 1)$$
$$\sigma \sim Uniform(0, 10)$$
$$S = [male, female]$$

```
data("Howell1")
d = Howell1 %>% subset(age < 13)

datq3 = list(W=d$weight, A=d$age,S=d$male+1)
cat_map = c('female','male')

m_q3 = quap(
    alist(
        W ~ dnorm(mu,sigma),
        mu <- a[S] + b[S]*A,
        a[S] ~ dnorm(4,2),
        b[S] ~ dlnorm(0,1),
        sigma ~ dunif(0,10)
    ),
    data=datq3
)
precis(m_q3,depth=2)
```

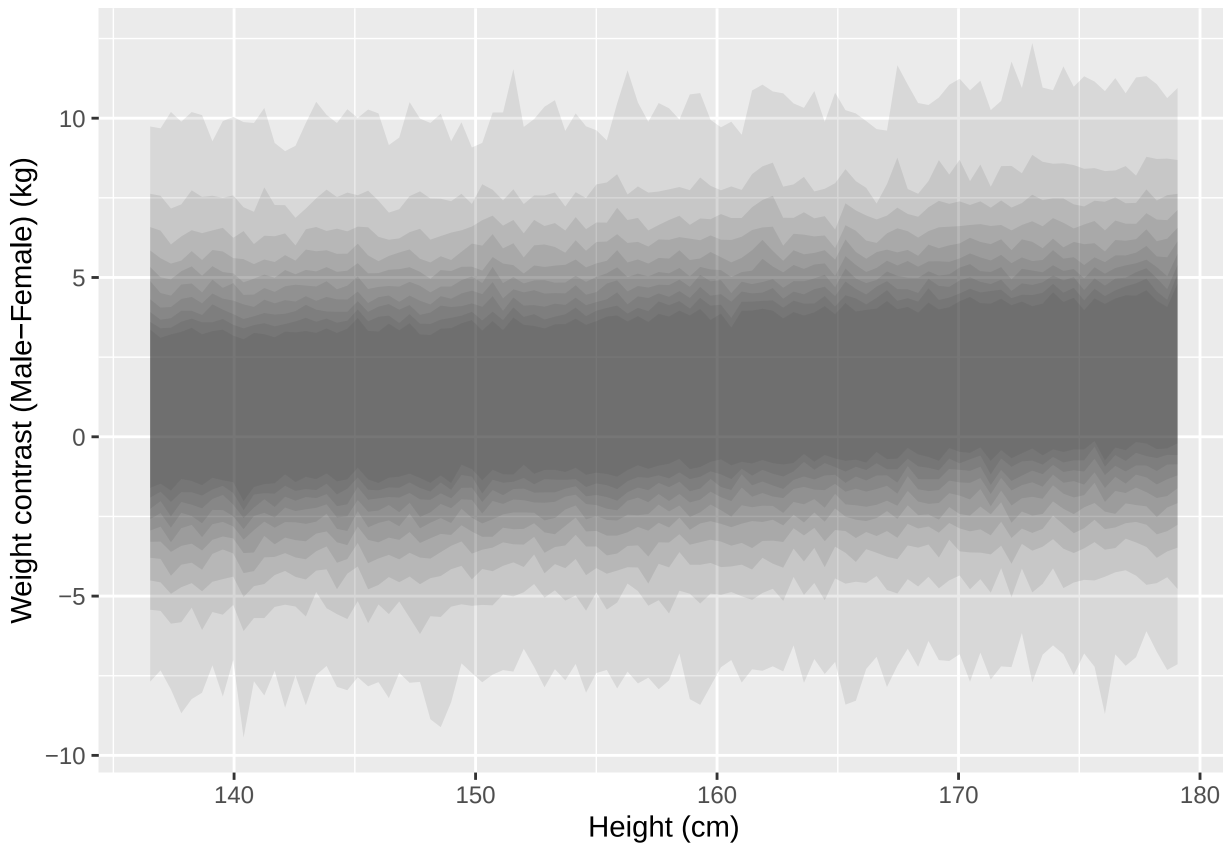
```
##              mean         sd     5.5%    94.5%
## a[1]  6.919221 0.46612000 6.174272 7.664171
## a[2]  7.680137 0.49227292 6.893390 8.466884
## b[1]  1.305670 0.07184250 1.190852 1.420489
## b[2]  1.412396 0.07449973 1.293331 1.531461
## sigma 2.425733 0.14218991 2.198487 2.652980
```

```
as = seq(from=min(d$age),to=max(d$age),len=100)

w_f = link(fit=m_q3,data=list(S=rep_along(hs,1),A=as))
w_m = link(fit=m_q3,data=list(S=rep_along(hs,2),A=as))
w_c = w_m - w_f

intervals = seq(from=0.5,to=0.99,len=10) %>%
    map_dfr(~apply(w_c,2,PI,prob=.x) %>%
                t %>%
                as_tibble() %>%
                set_names(c('ymin','ymax')) %>%
                mutate(hs=hs),.id='i')
```

```r
ggplot(intervals,aes(x=hs,ymin=ymin,ymax=ymax,group=i)) +
    geom_ribbon(alpha=0.1) +
    xlab('Height (cm)') +
    ylab('Mean weight contrast (Male-Female) (kg)')
```




```r
w_f = sim(fit=m_q3,data=list(S=rep_along(hs,1),A=as))
w_m = sim(fit=m_q3,data=list(S=rep_along(hs,2),A=as))
w_c = w_m - w_f

intervals = seq(from=0.5,to=0.99,len=10) %>%
    map_dfr(~apply(w_c,2,PI,prob=.x) %>%
                t %>%
                as_tibble() %>%
                set_names(c('ymin','ymax')) %>%
                mutate(hs=hs),.id='i')

ggplot(intervals,aes(x=hs,ymin=ymin,ymax=ymax,group=i)) +
    geom_ribbon(alpha=0.1) +
    xlab('Height (cm)') +
    ylab('Weight contrast (Male-Female) (kg)')
```



Lecture 5: Elemental Confounds

Fork

$A \longleftarrow B \longrightarrow C$

- Causes **phantom** association between A and C
- Stratification by B **removes** this association
- $A \not\!\perp\!\!\!\perp B$ and $A \perp\!\!\!\perp B \mid C$
- Also known as a —common cause—

Pipe

$A \longrightarrow B \longrightarrow C$

- Causes **indirect** association between A and C
- Stratification by B **removes** this association
- $A \not\!\perp\!\!\!\perp B$ and $A \perp\!\!\!\perp B \mid C$
- Also known as a —chain— or —mediator—

Collider

$A \longrightarrow B \longleftarrow C$

- **No** association between A and C
- Stratification by B **causes** association
- $A \perp\!\!\!\perp B$ and $A \not\!\perp\!\!\!\perp B \mid C$
- Also known as a —collider—

Descendant

$$\begin{tikzcd} A & B \arrow[l] \arrow[r] \arrow[d] & C \\ & D & \end{tikzcd}$$

- D inherits whatever association B is subject to

Lecture 6: Good and Bad Controls

So what if we have a more problematic problem. What if we wanted to know the direct effect of a grandparents education on a child, where parent and child share an unobserved confound.

$$\begin{tikzcd} G \arrow[r] \arrow[dr] & P \arrow[d] & U \arrow[l] \arrow[dl] \\ & C & \end{tikzcd}$$

Stratefying by parent would lead to removal of G through P but introduce association through the collider on P. Of course it depends on the confounding strength, the bias might not be that important, so it is a tradeoff.

The Process

1. Clearly state assumptions
2. Determine logical consequences
3. Test

By not being —clever— and instead using simple deductions the model can be understood, and most importantly verify and challenge your work.

Randomisation

If we perform randomisation, that is control for any effects on a variable, then we can remove links. For instance on

$$\begin{tikzcd} A \arrow[d] \arrow[dr] & \\ B \arrow[r] & C \end{tikzcd}$$

if we randomise on B we get left with

$$\begin{tikzcd} A \arrow[dr] & \\ B \arrow[r] & C \end{tikzcd}$$

Do Calculus

So how do we do this in general?

$$P(C|do(B)) = \sum_A P(C|B, A)P(A)$$

So stratification on B and A and then averaged over A will give us the pure effect.

Cheetah, Baboon, Gazelle System

$$\begin{tikzcd} C \arrow[d] \arrow[dr] & \\ B \arrow[r] & G \end{tikzcd}$$

In such a simple system where Baboons only eat Gazelles if no Cheetahs are present, we need to know the distribution of Cheetahs to determine the total effect of Baboons on Gazelles. To determine how to get this use do-calculus, which has the advantage of being a-priori to the funcational fits. If inference is possible just from do-calculus, then it is possible without any assumptions; so it is the preffered place to be. However the power of inference possible is often greater after assumptions, but the assumptions must be true.

Backdoor Criterion

To find variables to stratify to yield

$$P(Y|do(X))$$

we:

1. Idetify all paths linking $X \to Y$
2. Take the subset entering X (Backdoor Paths)
3. Find adjustment sets that close the backdoor paths

Example 1

$$\begin{tikzcd} & Z \arrow[dr] & \\ X \arrow[ur] \arrow[rr] \arrow[dr] & & Y \arrow[dl] \\ & C & \end{tikzcd}$$

We have 3 paths here from $X \to Y$, and need to adjust nothing.

Example 2

$$\begin{tikzcd}
A \arrow[r] \arrow[d] & Z \arrow[dl] \arrow[dr] & B \arrow[l] \arrow[d] \\
X \arrow[rr] & & Y \arrow[dl] \\
& C \arrow[ul] &
\end{tikzcd}$$

but want

$$\begin{tikzcd}
A \arrow[r] & Z & B \arrow[l] \arrow[d] \\
X \arrow[ur] \arrow[rr] & & Y \arrow[dl] \\
& C &
\end{tikzcd}$$

There are six paths, one of which $X \to A \to Z \to B \to Y$ which will open if we stratify by Z. In the end we condition by C and Z but then also on either A or B, with B being the best choice.

Good and Bad Controls

Additionally looking at the course recommended paper (Cinelli, Forney, and Pearl, n.d.).

One of the worst case offenders for bad controls is the —m-bias—

$$\begin{tikzcd}
u \arrow[r] \arrow[d] & Z & v \arrow[l] \arrow[d] \\
X \arrow[rr] & & Y
\end{tikzcd}$$

Stratification on Z will open a path $X \to u \to Z \to v \to Y$

Case Control Bias

$$X \longrightarrow Y \longrightarrow Z$$

Do not control Z; you will reduce available variance for $X \to Y$ through reduced variance in Y, the parts not explained by Z. Remember that the statisticatal inference does not know the difference between causal and non-causal relaitionships.

Precision Parasite

$$Z \longrightarrow X \longrightarrow Y$$

Again do no control for Z; you will still get the correct mean, but the variance is increased.

Bias Amplification

$$\begin{tikzcd} Z \arrow[d] & u \arrow[dl] \arrow[dr] & \\ X \arrow[rr] & & Y \end{tikzcd}$$

If you control for Z, everything gets worse, the bias from u is increased.

Table 2 Fallacy

Example taken from (Westreich and Greenland 2013)]

$$\begin{tikzcd} Smoking \arrow[dr] \arrow[drr] & & \\ & HIV \arrow[r] & Stroke \\ Age \arrow[uu] \arrow[ur] \arrow[urr] & & \end{tikzcd}$$

If we control all of the causal variables for our outcome, the coefficients for a fully linear model becomes

$$Normal(\alpha + \beta_H H + \beta_S S + \beta_A A, \sigma)$$

which closes all three backdoor paths. Then by marginalising over age and smoking, we can get the direct effect. From the perspective of S, the same regression is sratified on X, giving only the direct effect $S \rightarrow Stroke$. Simalarily for age, two pathways are closed, only getting the direct effects. This means that the coefficients in a table of this fit mean different things, and on it's own is useless. So what can we do?

1. Only provide non-control, marginalised as appropriate
2. Provide interpretation explicitly of all the coefficients

Week 3 Homework

Urban Foxes

We have the following model for measurements

$$\begin{tikzcd} & A \arrow[dl] & \\ F \arrow[rr] \arrow[dr] & & G \arrow[dl] \\ & W & \end{tikzcd}$$

between Area, Food (average), Groupsize, and Weight.

```
data(foxes)

foxes <- foxes %>%
    mutate(across(-any_of('group'),standardize)) %>%
    rename(F=avgfood, G=groupsize, A=area, W=weight)
```

Q1: Determine $A \to F$

```
atog <- quap(
    alist(
        F ~ dnorm( mu, sigma ),
        mu <- a + bA * A,
        a ~ dnorm(0,0.3),
        bA ~ dnorm(0,0.6),
        sigma ~ dexp(1)
    ),
    data=foxes
)
precis(atog)
```

```
##                 mean         sd        5.5%      94.5%
## a      -1.222322e-06 0.04284533 -0.06847633 0.06847389
## bA      8.784904e-01 0.04336736  0.80918100 0.94779983
## sigma   4.662377e-01 0.03052046  0.41746012 0.51501531
```

A: Very linear increase.

Q2: Total and Direct $F \to W$

```
ftow_full <- quap(
    alist(
        W~ dnorm( mu, sigma ),
        mu <- a + bF * F,
        a ~ dnorm(0,0.3),
        bF ~ dnorm(0,0.6),
        sigma ~ dexp(1)
    ),
    data=foxes
)
precis(ftow_full)
```

```
##                 mean         sd       5.5%     94.5%
## a       9.875573e-07 0.08797911 -0.1406066 0.1406086
## bF     -2.445827e-02 0.09134717 -0.1704487 0.1215322
```

```
## sigma  9.911430e-01 0.06465842  0.8878063 1.0944796
```

Need to close G pipe.

```
ftow_direct <- quap(
    alist(
        W~ dnorm( mu, sigma ),
        mu <- a + bF * F + bG * G,
        a ~ dnorm(0,0.3),
        bF ~ dnorm(0,0.6),
        bG ~ dnorm(0,0.6),
        sigma ~ dexp(1)
    ),
    data=foxes
)
precis(ftow_direct, depth=2)
```

```
##                mean         sd       5.5%      94.5%
## a      8.209217e-08 0.08386956 -0.1340397  0.1340398
## bF     5.185901e-01 0.18512758  0.2227205  0.8144597
## bG    -6.153381e-01 0.18513764 -0.9112238 -0.3194524
## sigma  9.408162e-01 0.06156220  0.8424279  1.0392045
```

Looks like constant wolves per average food. This might be a response measure as a consequence of having additional food supply (an intervention on area), the group gets larger. Other interpretations would change the causal nature of our model, group size intervention might affect the area the group has.

Q3 Table 2 Fallacy with unobserved variables

$$
\begin{tikzcd}
S \arrow[dr] \arrow[drr] & & u \arrow[ll] \arrow[d] \\
 & X \arrow[r] & Y \\
A \arrow[uu] \arrow[ur] \arrow[urr] & &
\end{tikzcd}
$$

The two backdoors will be closed by stratifying on S and A. This will cause a collider through S to open up, but that doesn't involve X. The interpretation of the other variables are now affected by u, which means that they no longer correspond to the direct effects as they did in the model without the unobserved variable.

Lecture 7: Overfitting

Again stress the difference between predictions and causal; description of the points and explain the points.

- It is very possible to get good predictions about what further observations would yield. Even if we do not properly understand the causes.

Leave-one-out Cross-validation

Take the L^2 sum of differences between fit prediction with each point in turn taken out of the fit. Then compare this to the same distances for the full fit.This gives us an understanding of how over/underfit our model is based on degree of freedom.

Log Pointwise Predictive Density (LPPD)

$$lppd_{CV} = \sum_{i}^{N} \frac{1}{S} \sum_{s}^{S} \log Pr(y_i|\theta_{-i,s})$$

for N data points and S samples of the posterior. We log for stability due tohardware represeta-tions for our models.

Regularisation

Cross-validation does not handle overfitting, it just improves choices between their model, not the model itself.

From one perspective overfitting can come from the parametric structure and also the prior. The priors determine flexibility of your models, **skeptical priors** are restricted priors that reduce the space your model can explore. Beware of underfitting with to restricted priors, but in general priors are more restricted than you would naievly think they are.

So how do we tune our priors, well that depends on what we are doing.

- For pure prediction, tune using your data
- For causal inference, use a priori from science knowledge

In reality there would be a mix of causal and prediction in the model. Remember no prior is perfect, you need to only be better than the unconstraining prior.

Importance Sampling

Well doing this leave one out fitting can get expensive fast so we use importance sampling to work on it post fit. The key take-away is that any data point with now prabability according to the model has a larger affect on the model fit than a typical point. Another way of thinking about this is that removing an outlier from a fit changes the posterior the most.

Naive importance sampling can have unreliable results, so instead we will use (one of many), Pareto-improved importance sampling (PSIS).

Altenatively is to use widely applicable information criteria (WAIC)

$$WAIC(y, \Theta) = -2(lppd - \sum_{i} var_{\Theta} \log Pr(y_i|\Theta)$$

The sum is the penalty term and in perfect normal land is just the degrees of freedom.

Both PSIS and WAIC perform remarkably similar, mut the former also has automated diagnostic checks.

Model Mis-selection

Neither of these things address anything about causal inference. They prefer confounds and colliders!

Outliers

Dropping outliers is bad; does not improve prediction. This is often due to differing distributions in different data points based on unmeasured parameters. By fitting to a student-t distrobution, which is a squished down gaussian, with larger tails. This is in effect fitting with multiple gaussians in aggregate, and can improve the precision of your fit.

Lecture 8: Markov Chain Monte Carlo

To drawing the Baysian owl, if your response —Just Analyse the data— is a bit sarcastic, your fairly justified. In simple cases we can just reason forms for the solutions, but life isn't simple. Lots of problems aren't multi variate Gaussians, so we need to expand our repetoir.

While MCMC is computatinally intensive, it has way more flexible. We can visit each parameter in proportion to it's probability, thus mapping through arbitrary parameter space. Need to only know relative probability of two spots at a time for the next step.

Metropolis was the first MCMC alogrithm but now gradient based methods are more in use, instead for instance Hamiltonian MC, so trajectories using pseudo momentum and potential to get weighted samples. So you need the derivatives of your parameters, or nearby points to estimate them.

Auto-diff

Automatically calculates derivatives from your statistical model for your gradient model giving you the Jacobian. STAN math libraries to the rescue.

STAN Code

- TODO: add stancode(x) output from homework

MC Owl

Due to the long research diagnostics are well matured.

Trace Plot

Plots timeline of each parameter as a timeline, to see whether parameter space was sampled nicely. Ie. no drifting or long term trend evident. To really test this use multiple independent chains and ensure they converge to the same distrobution, and this is trivially possible. We then layer the trajectories on the same trace plot.

Trace Rank Plot

Instead of parameter value use the rank instead. This shows whether any chain is on top of any others.

$\hat{R}$

The ratio of variance in chain against total variance. Good chains' variance ends up as the whole variance in the chain. Large values are bad and close to 1 is good.

n_{eff}

Takes account the autocorrelation, a read out of the effectiveness of your stepsize.

Divergent Transitions

While HMC makes good proposal, if the discrete simulation parameters add enough error, some proposals will still need to be rejected.

The Folk Theorem of Statistical Computing

When you have computational problems, often there's a problem with your model - Andrew Gelman

Lecture 9: Modelling Events

The learning example is influence of admission rates, with per department data. This leads to a very common basic mediating pathway:

$$\begin{tikzcd} & D \arrow[dr] & \\ G \arrow[ur] \arrow[rr] & & A \end{tikzcd}$$

Remember again the data itself does not contain any of the causes, so in these discrimination based research is difficult and requires carefull work.

Types of Discrimination

In the literature we divide direct discrimination

- **Status Based (statistical) Discrimination:** Not direct against knowledge of the category being discriminated
- **Taste Based Discrimination:** Direct causal effect from category

Then the mediating paths are indirect discriminations; **structual discrimination**. In our example even if each department has equal admission rates on gender, overall there could still be total discrimination.

GLM

Switching from a linear model to generalised linear models we go from

$$
\begin{aligned}
Y_i &\sim Normal(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_X X_i + \beta_Y Y_i
\end{aligned}
$$

to

$$
\begin{aligned}
Y_i &\sim Bernoulli(p_i, \sigma) \\
f(p_i) &= \alpha + \beta_X X_i + \beta_Y Y_i
\end{aligned}
$$

with some function f; the link function. So we can use this to restrict the probability to $[0, 1]$. Then

$$
p_i = f^{-1}(\alpha + \beta_X X_i + \beta_Y Y_i)
$$

Logit Link

Arrissing naturally from normal distributions, the logit function is way of mapping $[0, 1]$ to $\mathbb{R}$ without distortion. The logit function is just the log odds

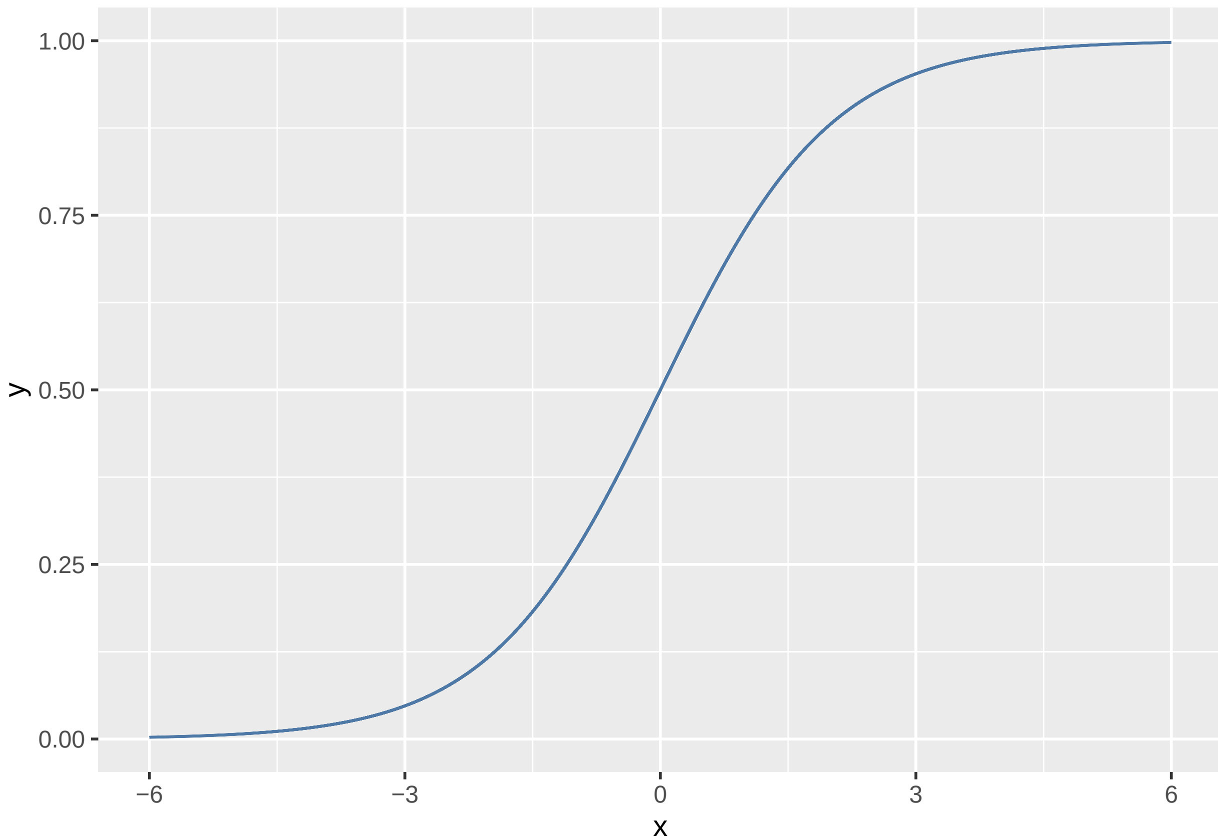
$$
logit(p_i) = \log \frac{p_i}{1 - p_i}
$$

which has the logistics function as inverse

$$
logit^{-1}(q_i) = \frac{e^{q_i}}{1 + e^{q_i}}
$$

In practice this works really well, which is the real reason we use it. It is then fairely easy to read log odd values as really $logit(6) \approx 1$ and $logit(0) = 0.5$.

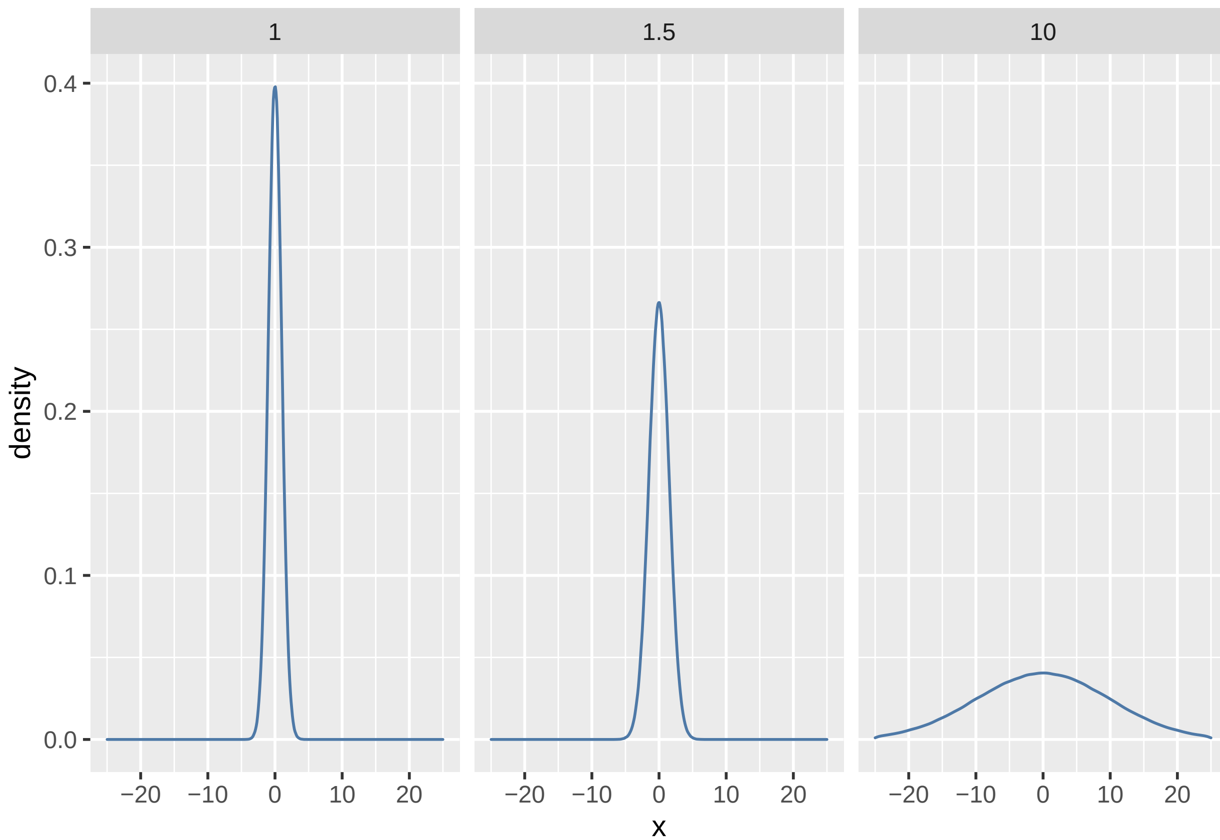
```
x_bound <- 6
df   <- data.frame( x=seq(-x_bound,x_bound,length.out=1e6) )
df$y <- inv_logit(df$x)

ggplot(df) + geom_line(aes(x=x,y=y),colour=ggthemes::tableau_color_pal()(1))
```



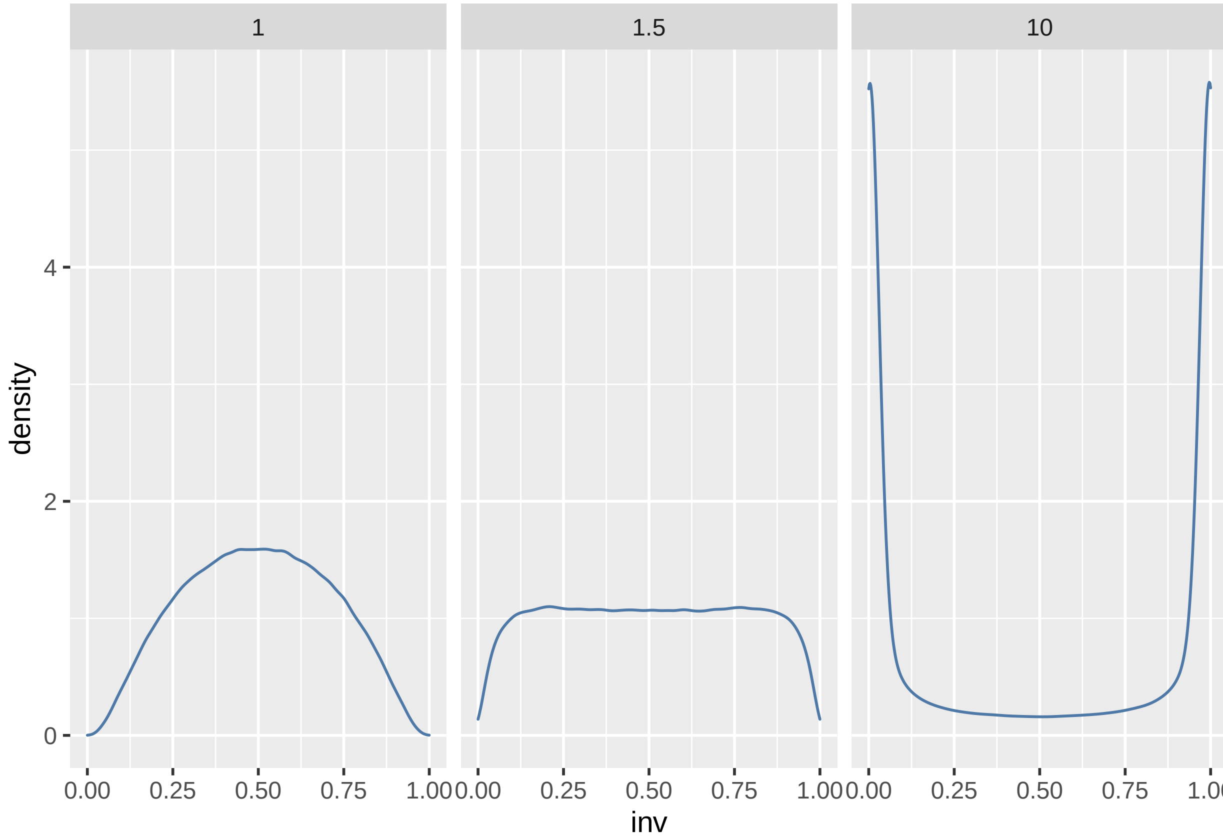
Now the question arises as to good priors starting from the simplest case, constant value.

```
samples <- 1e6
sigmas <- c(10,1.5,1)
df <- map_dfr(set_names(sigmas,sigmas), function(s) data.frame(x=rnorm(samples,sd=s)), .id=
    mutate(inv = inv_logit(x))

ggplot(df,aes(x=x)) +
    geom_density(colour=ggthemes::tableau_color_pal()(1)) +
    facet_grid(~sigma) +
    xlim(-25,25)
```

```
## Warning: Removed 12237 rows containing non-finite values (stat_density).
```




```r
ggplot(df,aes(x=inv)) +
    geom_density(colour=ggthemes::tableau_color_pal()(1)) +
    facet_grid(~sigma)
```



So a reasonable flat distribution is $\sigma = 1.5$ and a distribution that emphasizes non extreme values $\sigma = 1$. Of course the large sigma strongly favours extreme results.

Stan matrix notation

```
alist(
    A ~ bernoulli(p),
    logit(p) <- a[G,D],
    matrix[G,D]:a ~ normal(0,1)
)
```

Binomial Regression

Depending on data structure the equiavlent binmial regression to

$$A_i \sim Bernoulli(p_i)$$
$$logit(p_i) = \alpha[G_i, D_i]$$

is

$$A_i \sim Binomial(N_i, p_i)$$
$$logit(p_i) = \alpha[G_i, D_i]$$

moving $[0,1] \to [0...N]$.

Marginal Causal Effect

Now beware, when we perform an intervention on gender, we are really changing the percieved gender $G \to P \to A$. This can have subtle implications on what the question we are actually answering.

Beware

Discrimination effects can hide in all sorts of places, for instance from the department choice itself; here be confounds.

Week 4 Homework

Q1: Marriage Age and Happiness

$$\begin{tikzcd}
 & A \arrow[dl] \arrow[dr] & \\
M \arrow[rr] & & H
\end{tikzcd}$$

```r
library(cmdstanr)
df <- sim_happiness() %>%
    subset(age >= 18) %>%
    mutate( M = married + 1
          , A = (age-18)/(max(age)-18)
          , H = happiness )
```

```r
a6.9 <- alist( H ~ dnorm(mu,sigma)
             , mu <- a[M] + bA * A
             , a[M]  ~ dnorm(0,1)
             , bA    ~ dnorm(0,2)
             , sigma ~ dexp(1) )
m6.9m <- ulam(a6.9, data=df, chains=4, cores=2)
```

```
## Warning in '/tmp/Rtmpwvp5KC/model-7d668f5a1f4.stan', line 3, column 4: Declaration
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
## Warning in '/tmp/Rtmpwvp5KC/model-7d668f5a1f4.stan', line 4, column 4: Declaration
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
```

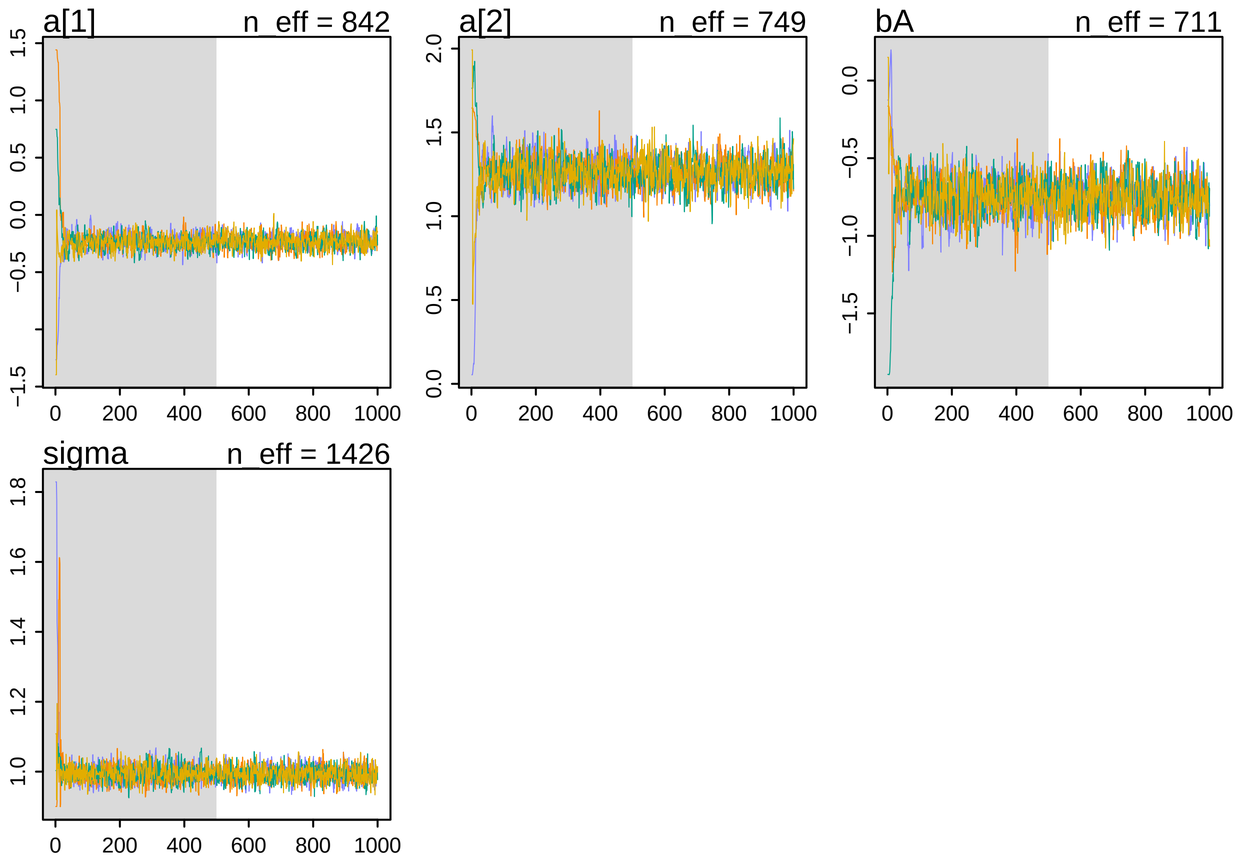
```
## Warning in '/tmp/Rtmpwvp5KC/model-7d668f5a1f4.stan', line 7, column 4: Declaration
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc

## Running MCMC with 4 chains, at most 2 in parallel, with 1 thread(s) per chain...
##
## Chain 1 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1 Iteration: 501 / 1000 [ 50%]  (Sampling)

## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 1 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/tmp/Rt

## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 1 but if this warning occurs often then your model may be either severely ill-cond

## Chain 1

## Chain 2 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1 finished in 0.2 seconds.
## Chain 2 finished in 0.2 seconds.
## Chain 3 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3 Iteration: 300 / 1000 [ 30%]  (Warmup)
```

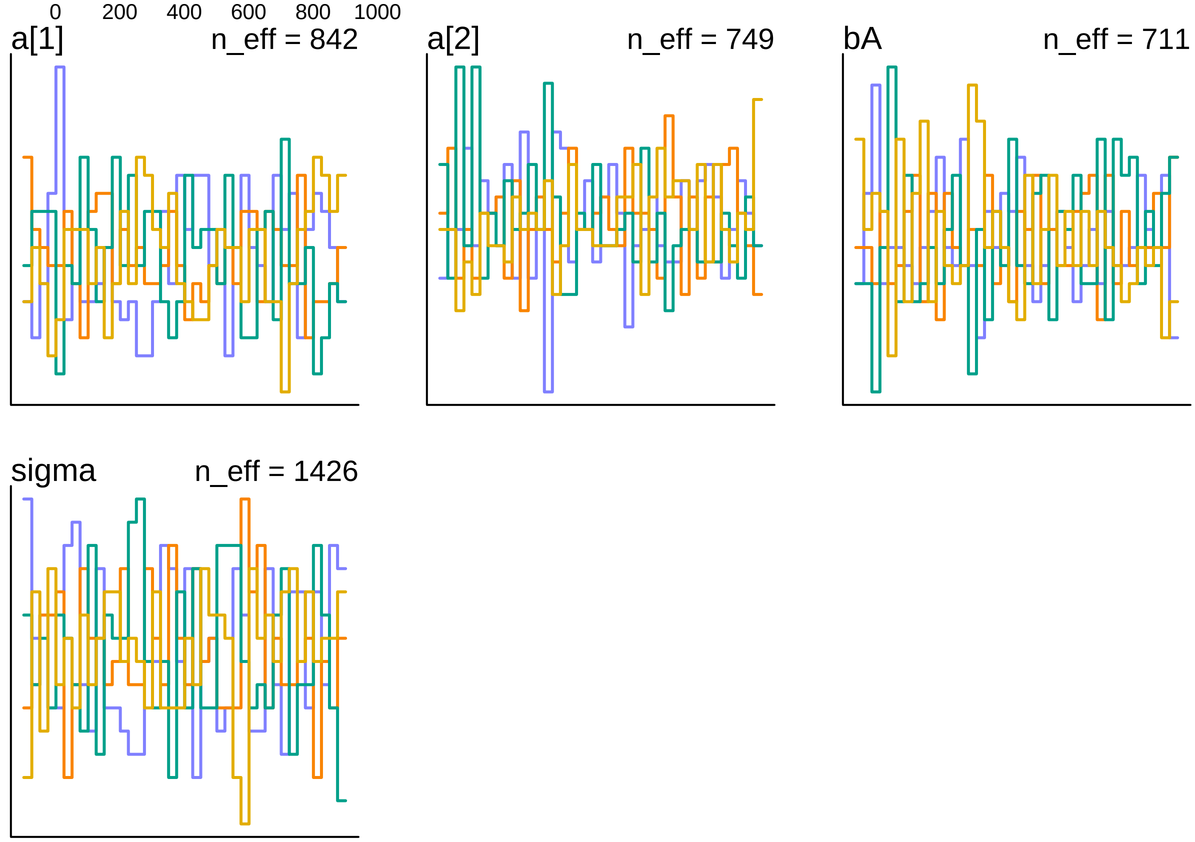
```
## Chain 3 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3 finished in 0.2 seconds.
## Chain 4 finished in 0.2 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.2 seconds.
## Total execution time: 0.5 seconds.
```

```r
m6.9q <- quap(a6.9, data=df)
precis(m6.9m,depth=2)
```

```
##             mean         sd       5.5%      94.5%     n_eff    Rhat4
## a[1]  -0.2316120 0.06362552 -0.3327085 -0.1312628  841.7176 1.005407
## a[2]   1.2632410 0.08564636  1.1282156  1.3990471  748.7855 1.006312
## bA    -0.7573706 0.11350242 -0.9370262 -0.5742688  710.8936 1.005853
## sigma  0.9926699 0.02225567  0.9583219  1.0293333 1425.8814 0.999023
```

```r
traceplot_ulam(m6.9m)
trankplot(m6.9m)
```






```
a6.10 <- alist( H ~ dnorm(mu,sigma)
              , mu <- a + bA * A
```

```
             , a     ~ dnorm(0,1)
             , bA    ~ dnorm(0,2)
             , sigma ~ dexp(1) )
m6.10m <- ulam(a6.10, data=df, chains=4, cores=2)
```

```
## Warning in '/tmp/Rtmpwvp5KC/model-7d6650414d44.stan', line 2, column 4: Declaration
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
## Warning in '/tmp/Rtmpwvp5KC/model-7d6650414d44.stan', line 4, column 4: Declaration
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
## Warning in '/tmp/Rtmpwvp5KC/model-7d6650414d44.stan', line 5, column 4: Declaration
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
```

```
## Running MCMC with 4 chains, at most 2 in parallel, with 1 thread(s) per chain...
##
## Chain 1 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2 Iteration: 900 / 1000 [ 90%]  (Sampling)
```

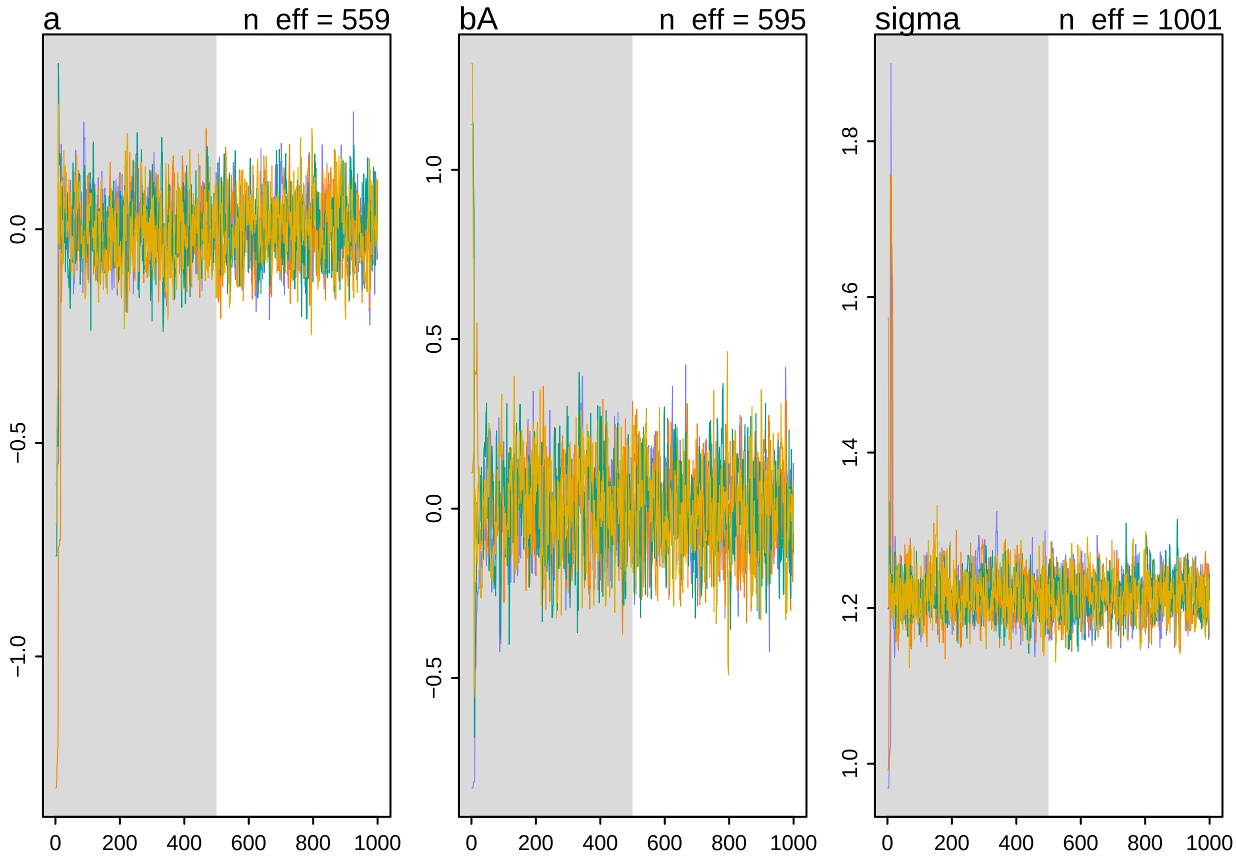
```
## Chain 2 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1 finished in 0.2 seconds.
## Chain 2 finished in 0.2 seconds.
## Chain 3 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4 Iteration: 600 / 1000 [ 60%]  (Sampling)

## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 4 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/tmp/R

## Chain 4 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 4 but if this warning occurs often then your model may be either severely ill-cond

## Chain 4

## Chain 3 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3 finished in 0.2 seconds.
## Chain 4 finished in 0.2 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.2 seconds.
## Total execution time: 0.4 seconds.
```

```r
m6.10q <- quap(a6.10, data=df)
precis(m6.10m,depth=2)
```

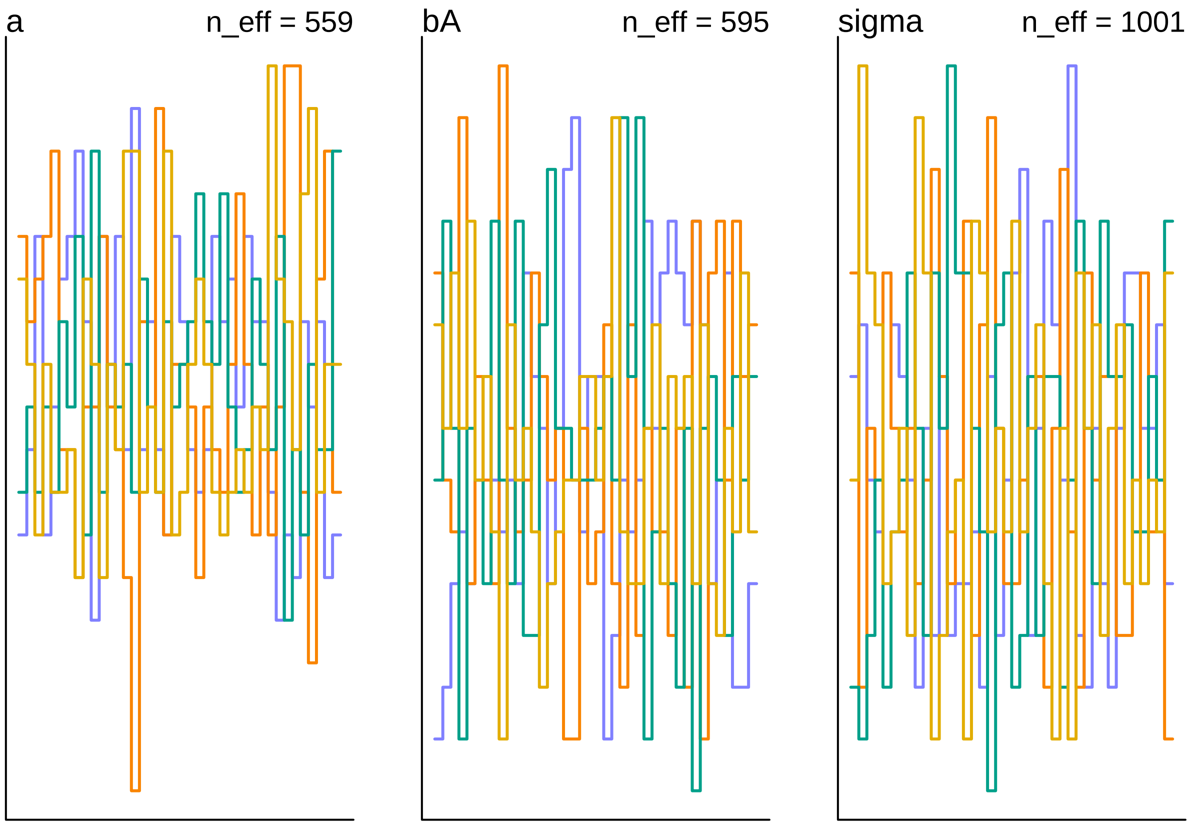
```
##               mean        sd      5.5%     94.5%     n_eff     Rhat4
```

```
## a       0.002361546 0.07850328 -0.1216828 0.1296681  558.7915 0.9994368
## bA     -0.003716437 0.13336564 -0.2210628 0.2070303  595.2580 0.9988932
## sigma   1.216046375 0.02813230  1.1710578 1.2613700 1001.1836 1.0018885
```

```
traceplot_ulam(m6.10m)
```




```
trankplot(m6.10m)
```




```r
compare(m6.9q, m6.10q, func=WAIC)
```

```
##            WAIC       SE    dWAIC      dSE    pWAIC       weight
## m6.9q  2713.459 37.49213   0.0000       NA 3.464575 1.000000e+00
## m6.10q 3101.958 27.78053 388.4998 35.40747 2.373800 4.348588e-85
```

```r
compare(m6.9q, m6.10q, func=PSIS)
```

```
##            PSIS       SE    dPSIS      dSE    pPSIS       weight
## m6.9q  2713.546 37.48510   0.0000       NA 3.516314 1.000000e+00
## m6.10q 3102.058 27.74153 388.5124 35.30064 2.432978 4.321227e-85
```

Answer WAIC and PSIS both say that the first model is better, but it is clear from the DAG that stratification on age opens a non causal path through age.

Q2: Urban Foxes Revisited

```r
data(foxes)

foxes <- foxes %>%
    mutate(across(-any_of('group'),standardize)) %>%
    rename(F=avgfood, G=groupsize, A=area, W=weight)

ftow_full <- quap(
    alist(
```

```
        W~ dnorm( mu, sigma ),
        mu <- a + bF * F,
        a ~ dnorm(0,0.3),
        bF ~ dnorm(0,0.6),
        sigma ~ dexp(1)
    ),
    data=foxes
)
ftow_direct <- quap(
    alist(
        W~ dnorm( mu, sigma ),
        mu <- a + bF * F + bG * G,
        a ~ dnorm(0,0.3),
        bF ~ dnorm(0,0.6),
        bG ~ dnorm(0,0.6),
        sigma ~ dexp(1)
    ),
    data=foxes
)

compare(ftow_full, ftow_direct, func=WAIC)
```

```
##                 WAIC       SE    dWAIC      dSE    pWAIC      weight
## ftow_direct 323.8558 16.25632 0.000000       NA 3.862662 0.993109111
## ftow_full   333.7970 13.80556 9.941281 7.039553 2.600237 0.006890889
```

```
compare(ftow_full, ftow_direct, func=PSIS)
```

```
##                 PSIS       SE    dPSIS     dSE    pPSIS     weight
## ftow_direct 324.4066 16.54362 0.000000      NA 4.166728 0.98995944
## ftow_full   333.5887 13.88619 9.182062 7.24709 2.486485 0.01004056
```

There is no real difference between the two, but a represents the normal weight for the wolf, and β_F the total change in weight from a hypothetical intervention on food.

Q3: Cherry Blossom Precition

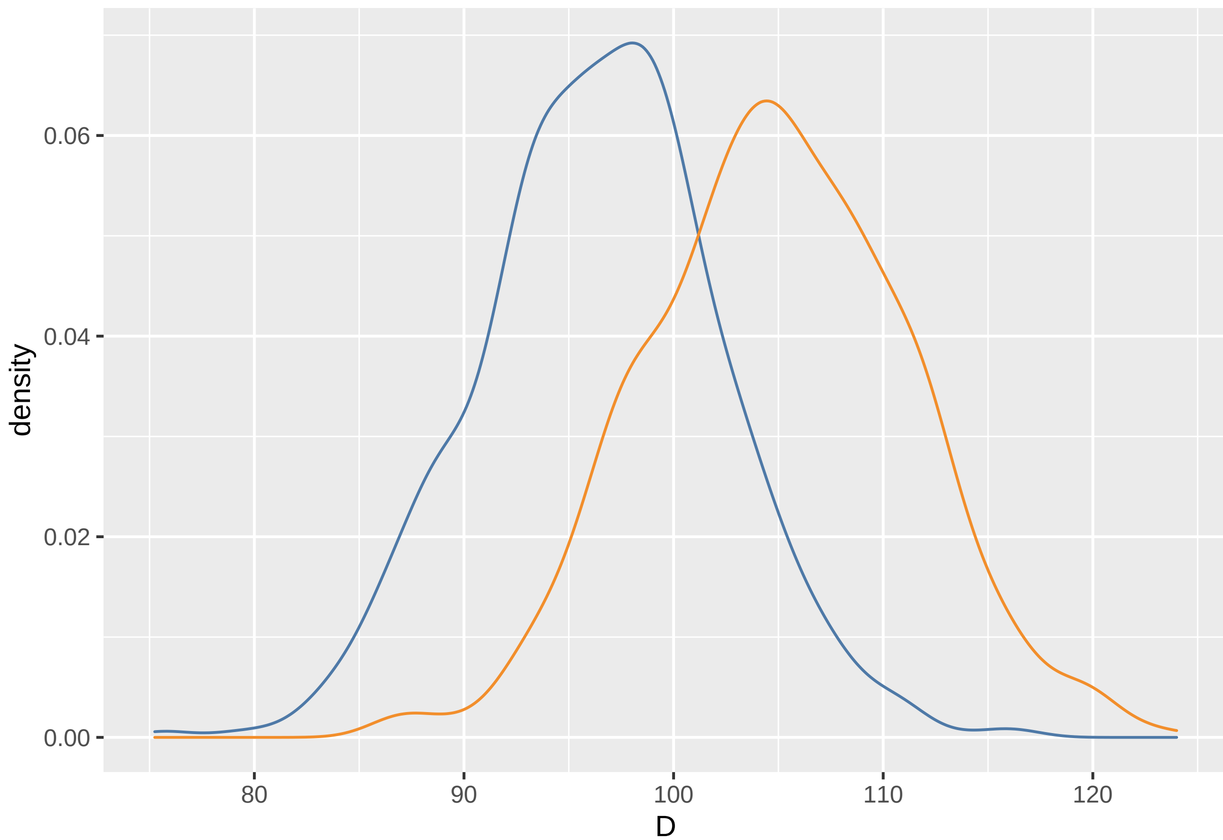
$$Y \longrightarrow T \longrightarrow D$$

```
data(cherry_blossoms)
df_raw <- cherry_blossoms %>%
    select(year,doy,temp) %>%
    subset(complete.cases(.)) %>%
    rename(Y=year, T=temp, D=doy)
df <- mutate(df_raw, across(-any_of('Y'),standardize))
```

```r
a_const <- alist( D ~ dnorm(mu, sigma)
                , mu <- a
                , a     ~ dnorm(0,1)
                , sigma ~ dexp(1) )
a_linear <- alist(  D ~ dnorm(mu, sigma)
                 , mu <- a + bT * T
                 , a     ~ dnorm(0,1)
                 , bT    ~ dnorm(0,1)
                 , sigma ~ dexp(1) )
a_quadratic <- alist(  D ~ dnorm(mu, sigma)
                    , mu <- a + bT * T + bT2 * T**2
                    , a     ~ dnorm(0,1)
                    , bT    ~ dnorm(0,1)
                    , bT2   ~ dnorm(0,1)
                    , sigma ~ dexp(1) )
a_cubic <- alist(  D ~ dnorm(mu, sigma)
                , mu <- a + bT * T + bT2 * T**2 + bT3 * T**3
                , a     ~ dnorm(0,1)
                , bT    ~ dnorm(0,1)
                , bT2   ~ dnorm(0,1)
                , bT3   ~ dnorm(0,1)
                , sigma ~ dexp(1) )

m_const     <- quap(a_const    , data=df)
m_linear    <- quap(a_linear   , data=df)
m_quadratic <- quap(a_quadratic, data=df)
m_cubic     <- quap(a_cubic    , data=df)

compare(m_const, m_linear, m_quadratic, m_cubic, func=PSIS)
```

```
##                 PSIS       SE     dPSIS        dSE    pPSIS       weight
## m_linear    2149.145 40.97735  0.000000         NA 2.789009 6.460524e-01
## m_quadratic 2151.163 40.89331  2.018239  0.2140600 3.657231 2.355118e-01
## m_cubic     2152.538 40.87859  3.393021  0.9933681 4.469429 1.184357e-01
## m_const     2236.469 39.60190 87.324269 16.8287709 2.021384 7.047638e-20
```

```r
temperature <- 9
z <- (temperature - mean(df_raw$T))/sd(df_raw$T)

res <- data.frame(doy_p=sim( m_linear, data=list(T=z))) %>%
    mutate(D = doy_p * sd(df_raw$D) + mean(df_raw$D))

ggplot(res,aes(x=D)) +
    geom_density(colour=ggthemes::tableau_color_pal()(1)[1], group='Sim') +
    geom_density(colour=ggthemes::tableau_color_pal()(2)[2], group='Dat', data=df_raw)
```



A

predicted earlier bloom.

Lecture 10: Counts and Confounds

$$\begin{tikzcd}
 & D \arrow[dr] & u \arrow[l] \arrow[d] \\
G \arrow[ur] \arrow[rr] & & A
\end{tikzcd}$$

Looking at admission based on gender and department with unobserved skill u. We can get the total affect of gender on admissions but not the direct affect due to the collider on D. The effect of this confound can mask the effect of discrimination, people making choices based on their skill level and their knowledge of departments discrimination.

If we had access to u we could of course remove the confound by stratifying on u as well and everything becomes alright. But in practice how do we get around it. Ideally would randomise the department applied to, but not really practically possible in every case. So what are the other options:

1. Sensitivity Analysis
2. Proxies

Sensitivity Analysis

Trying to determine consequence of confound based on strength of the confound. In other words the question we answer is: how strong must the confound be to affect our answer. Simply put add it to the model and instead of passing in, add models for each of it's effects and pass in the coefficients for it.

Proxies

But what if we could observe some other related quantites, for instance some test scores.

$$\begin{tikzcd}
 & & T1 & \\
 & D \arrow[dr] & u \arrow[l] \arrow[u] \arrow[r] \arrow[d] & T2 \\
G \arrow[ur] \arrow[rr] & & A &
\end{tikzcd}$$

This would give us a model

$$\begin{aligned}
A_i &\sim Bernoulli(p_i) \\
logit(p_i) &= \alpha[G_i, D_i] + \beta_{G_i} u_i \\
u_k &\sim Normal(0, 1) \\
T_{ij} &\sim Normal(\mu_i, \tau_j)
\end{aligned}$$

We can then just fit for everything simultaneously.

Note

This model has more parameters than observations! This is possible because the relationship determines how many parameters you have, an these restrictions dramatically reduce your — effective— parameters.

Tools in Oceanic Societies

$$\begin{tikzcd}
C \arrow[dr] & L \arrow[l] \arrow[d] \arrow[dl] \\
P \arrow[u] \arrow[r] & T
\end{tikzcd}$$

Tools based on population, count and location. As there is no physical bound on the number of tools, this is a Poisson distribution, which has typical link function being log (log-linear models). This enforces positivity. However beware of good priors, $Normal(3, 0.5)$ is a good prior with about mean 20, so adjust as apropiate. Linear coefficients for such an intercept would be around $Normal(0, 0.2)$.

```
data(Kline)
```

$$T_i \sim Poisson(\lambda_i)$$
$$log(\lambda_i) = \alpha_{C_i} + \beta_{C_i} log(P_i)$$
$$\alpha_j \sim Normal(3, 0.5)$$
$$\beta_j \sim Normal(0, 0.2)$$

Here population is log normalised as it's suspected to be some diminishing returns.

Evolving the Fit

The naive fit does not have some physical expected inferences. At high population, high contact islands would have fewer tools than those with no contact. Aditionally at zero population this can predict finite tools. Either

1. Either use a more robust model, the student-t equivalent being the gamma-Poisson (negative-binomial)
2. Use scienctificly reasoned model

Innovation Loss Model

Start modelling change per unit time

$$\Delta T = \alpha_C P^{\beta_C} - \gamma T$$

Which models innovation rate α, elasticity β (dimminishing return) and per tool loss rate γ. So for equilibrium

$$\hat{T} = \frac{\alpha_C P^{\beta_C}}{\gamma}$$

This is the expected average amount of tools (in λ in our previous model).

Week 5 Homework

Q1: NWOGrants

```
data(NWOGrants)
df_raw <- NWOGrants
df <- df_raw %>%
    mutate(across(everything(),as.integer)) %>%
    rename(D=discipline, G=gender, N=applications, A=awards)
```

```r
a_total <- alist(
    A ~ dbinom(N,p),
    logit(p) <- a[G],
    a[G] ~ dnorm(0,1)
)

a_direct <- alist(
    A ~ dbinom(N,p),
    logit(p) <- a[G,D],
    matrix[G,D]:a ~ dnorm(0,1)
)

m_total  <- ulam(a_total , df, chains=2, cores=2, log_lik=TRUE)
```

```
## Warning in '/tmp/RtmpwvpKC/model-7d66538a7ff5.stan', line 2, column 4: Declaration
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
## Warning in '/tmp/RtmpwvpKC/model-7d66538a7ff5.stan', line 3, column 4: Declaration
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
## Warning in '/tmp/RtmpwvpKC/model-7d66538a7ff5.stan', line 4, column 4: Declaration
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
## Warning in '/tmp/RtmpwvpKC/model-7d66538a7ff5.stan', line 5, column 4: Declaration
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
```

```
## Running MCMC with 2 parallel chains, with 1 thread(s) per chain...
##
## Chain 1 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1 Iteration: 600 / 1000 [ 60%]  (Sampling)
```

```
## Chain 1 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.1 seconds.
```

```
m_direct <- ulam(a_direct, df, chains=2, cores=2, log_lik=TRUE)
```

```
## Warning in '/tmp/Rtmpwvp5KC/model-7d663aa40de1.stan', line 2, column 4: Declaration
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
## Warning in '/tmp/Rtmpwvp5KC/model-7d663aa40de1.stan', line 3, column 4: Declaration
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
## Warning in '/tmp/Rtmpwvp5KC/model-7d663aa40de1.stan', line 4, column 4: Declaration
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
## Warning in '/tmp/Rtmpwvp5KC/model-7d663aa40de1.stan', line 5, column 4: Declaration
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
```

```
## Running MCMC with 2 parallel chains, with 1 thread(s) per chain...
```

```
##
## Chain 1 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.1 seconds.
```

```r
precis(m_total , depth=2)
```

```
##           mean         sd      5.5%     94.5%    n_eff     Rhat4
## a[1] -1.738585 0.08288976 -1.875238 -1.609166 576.1683 0.9994469
## a[2] -1.529700 0.06585075 -1.633226 -1.424273 695.2024 0.9982276
```

```r
precis(m_direct, depth=3)
```

```
##              mean        sd      5.5%       94.5%    n_eff     Rhat4
## a[1,1] -0.9651537 0.3479257 -1.534221 -0.42422950 1789.456 0.9999197
## a[1,2] -1.7095577 0.2457432 -2.096024 -1.32696070 1491.527 0.9988874
## a[1,3] -1.3949083 0.1866136 -1.698117 -1.10357885 1304.754 0.9980822
## a[1,4] -1.2029121 0.2707401 -1.668260 -0.78707088 1186.732 0.9993831
## a[1,5] -2.0185848 0.1864100 -2.320174 -1.72109930 1728.898 0.9982206
## a[1,6] -1.1034368 0.3445027 -1.666265 -0.53771377 1211.028 0.9995831
```

```
## a[1,7] -0.8166725 0.5687151 -1.769378  0.04852449 1400.552 0.9997650
## a[1,8] -1.9984279 0.1610840 -2.270790 -1.74035385 1674.228 0.9999666
## a[1,9] -1.2278571 0.2860571 -1.693017 -0.78965314 1546.072 0.9987363
## a[2,1] -0.9804112 0.2168929 -1.334671 -0.65063108 1468.192 0.9986252
## a[2,2] -1.1104607 0.1978714 -1.434908 -0.80479242 1504.892 0.9988641
## a[2,3] -1.7405750 0.1767179 -2.017071 -1.47491395 1783.836 0.9980463
## a[2,4] -1.9022373 0.2747479 -2.339204 -1.47437835 1799.947 0.9981799
## a[2,5] -1.4346781 0.1569462 -1.686884 -1.19769505 1692.237 0.9980526
## a[2,6] -1.3821621 0.2005862 -1.704412 -1.06792775 1529.806 0.9991852
## a[2,7] -0.9452387 0.2717783 -1.400502 -0.53377133 1370.190 0.9993673
## a[2,8] -1.6906664 0.1389993 -1.908731 -1.48315770 1980.950 0.9982504
## a[2,9] -1.6237084 0.2012161 -1.954564 -1.31365440 2144.547 0.9987791
```

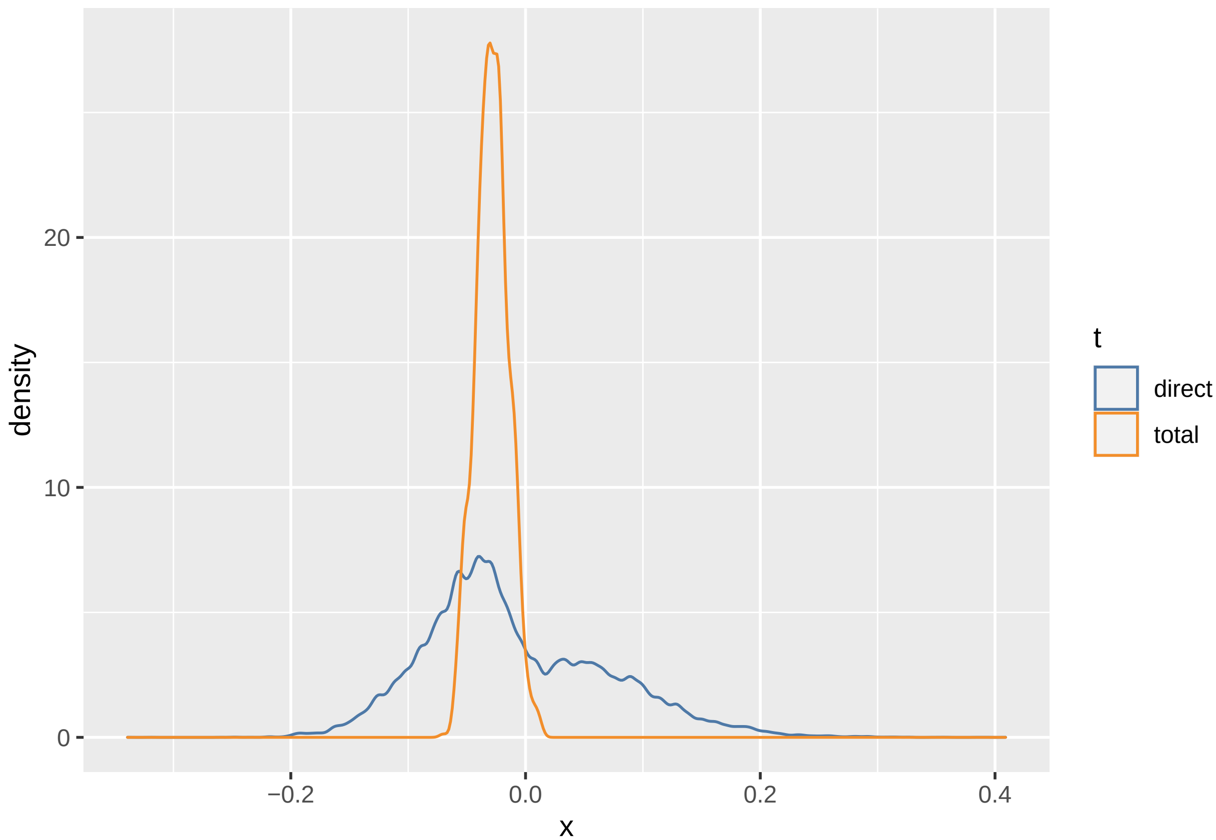
```r
compare(m_total, m_direct, func=PSIS)
```

```
## Some Pareto k values are high (>0.5). Set pointwise=TRUE to inspect individual points.
```

```
## Some Pareto k values are very high (>1). Set pointwise=TRUE to inspect individual points
```

```
##              PSIS       SE    dPSIS      dSE     pPSIS     weight
## m_direct 124.2141 5.184012 0.000000       NA 13.814635 0.95195881
## m_total  130.1871 9.134841 5.972926 7.609728  5.024821 0.04804119
```

```r
post_total <- extract.samples(m_total)
post_total$ia <- inv_logit(post_total$a)
total_contrast <- post_total$ia[,1] - post_total$ia[,2]

applicant_counts <- df %>% group_by(D) %>% summarise(N = sum(N))
total_applicants <- sum(df$N)
post_total_1 <- link(m_direct,data=list(
    D <- rep(applicant_counts$D,times=applicant_counts$N),
    N <- rep(1,total_applicants),
    G <- rep(1,total_applicants)
)) %>% as.vector()
post_total_2 <- link(m_direct,data=list(
    D <- rep(applicant_counts$D,times=applicant_counts$N),
    N <- rep(1,total_applicants),
    G <- rep(2,total_applicants)
)) %>% as.vector()
direct_contrast <- post_total_1 - post_total_2

data = rbind(data.frame(x=total_contrast ) %>% mutate(t='total')
            ,data.frame(x=direct_contrast) %>% mutate(t='direct'))

ggplot(data, aes(x=x, colour=t)) +
    geom_density() +
    scale_color_tableau()
```



Lecture 11: Ordered Categories

The examples in statistics courses are very simple. Of course in reality we meet real terror.

Trolley Problems

$$\begin{tikzcd}
X \arrow[r] & R & S \arrow[l] \\
E \arrow[ur] & A \arrow[l] \arrow[u] & G \arrow[ul] \arrow[ll, bend left]
\end{tikzcd}$$

Response R of trolley problem story S, with affecting variables education E, Age A and Gender G. Now imagine people respond on a scale 1 to 7, obviously 3 and 4 are closer than 3 and 6. Moreover each person has a different interpretation of the scale, but their own personal anchor around which they answer.

Selection Confound Participation

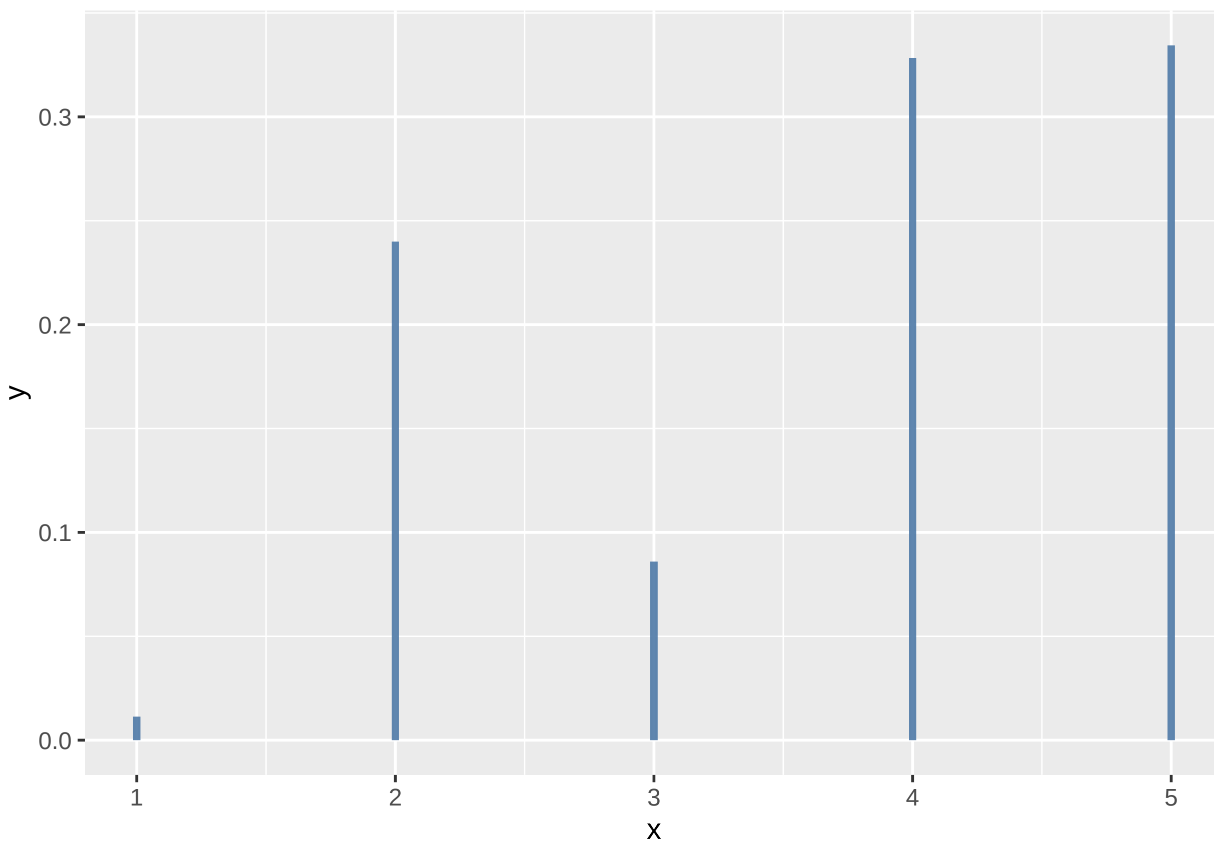
$$\begin{tikzcd}
X \arrow[r] & R & S \arrow[l] \\
E \arrow[ur] \arrow[dr] & A \arrow[l] \arrow[u] \arrow[d] & G \arrow[ul] \arrow[dl] \arrow[ll, bend left] \\
& P &
\end{tikzcd}$$

So backdoor paths have been opened for instance E, P, G, R. Of course in the study they knew that there was such problems and they did repeated measurements and multiple stories with the same structure to help deconvolude some of it.

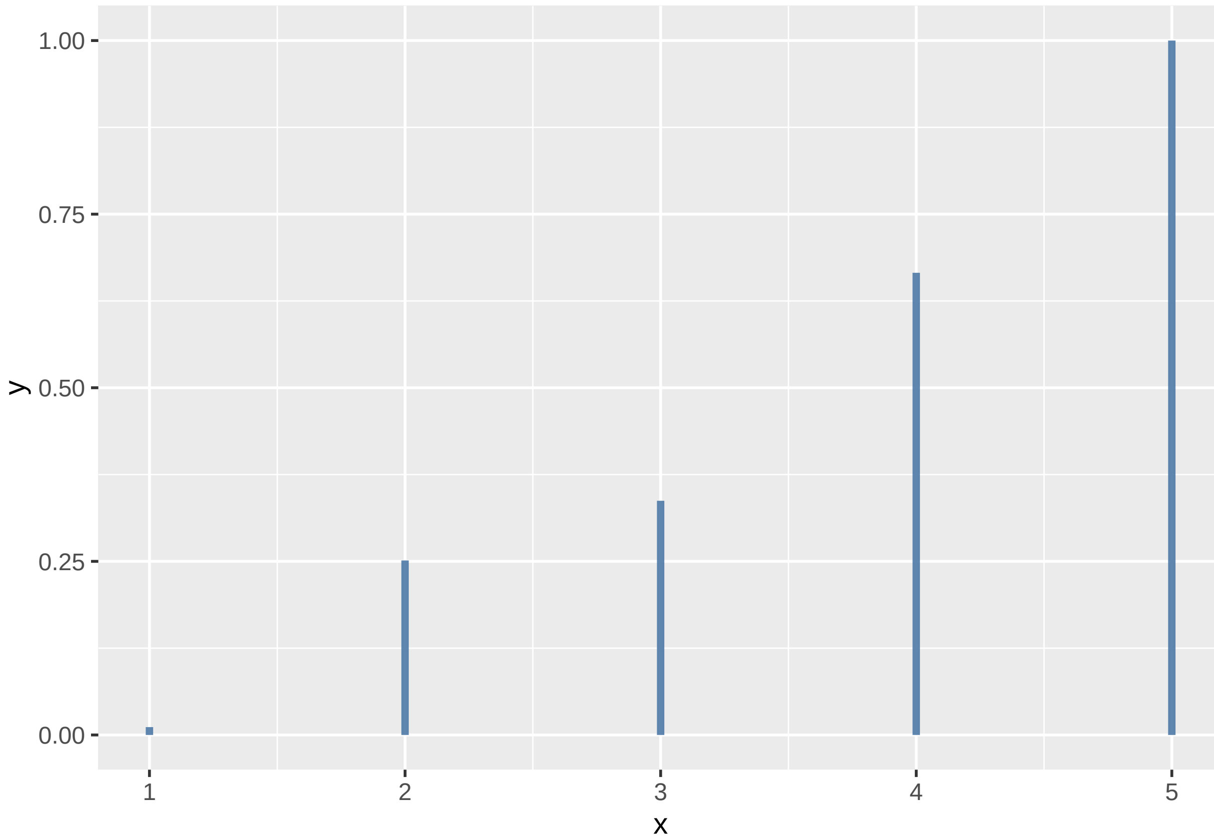
Ordered Logit

So how does one model the data with a natural order? Well us cumulative sum of a set, constructing the order.

```
xs <- rdirichlet(1, rep(2,5))[1,]
df <- data.frame(y=xs, x=seq_along(xs))
ggplot(df, aes(x=x, xend=x, y=0, yend=y, colour='a')) +
    geom_segment(size=1.3, alpha=0.9) +
    ggthemes::scale_color_tableau() +
    theme(legend.position = "none", panel.border = element_blank())
```



```r
ggplot(df %>% mutate(y = cumsum(y)), aes(x=x, xend=x, y=0, yend=y, colour='a')) +
    geom_segment(size=1.3, alpha=0.9) +
    ggthemes::scale_color_tableau() +
    theme(legend.position = "none", panel.border = element_blank())
```



We then fit for the horizontal cut-points, for instance on the log odds. So how does one add other variables to change these cuttpoints based on other variables? We use the ordered logit which has $n-1$ intercept parameters. The intercept parameters only encode the separation between intercepts and the other variables the anchor.

Ordered Predictors

So how does one have a monotonic affector variable (ie. Education level)

$$\phi_i = \beta_E \sum^{E_i} \delta_j$$

where

$$\sum \delta_j = 1$$

and

$$\delta_0 = 0.$$

Lecture 12: Multi Level Models

Revisiting the trolly problem

$$\begin{tikzcd}
& & U \arrow[dl] \\
X \arrow[r] & R & S \arrow[l] \\
E \arrow[ur] & A \arrow[l] \arrow[u] & G \arrow[ul] \arrow[ll, bend left]
\end{tikzcd}$$

now with individual U. So how does one add memory to this model?

Partial Pooling

For intance in our case the individual might have their own preference, simply replace the fixed σ in a prior to a fit prior. (exp(1) is a good prior for such a distribution). The fit σ then represents the memory in observations.

Note

- Fitting for σ adds dependecies for your other priors, reducing flexibility, in other words the effective parameters.
- Adding new (correctly identified causal) variables to the model will also reduce the fit σ.
  - By adding treatments one by one we can observe things about the size of effects we see, remember the highly non linear effect of parameters in our GLMs.

The Three Great Superstitions

- Different levels **do not** need to be sampled at random
- **Do not** need large sample sizes
- This **does not** assume Gaussian variation

Lecture 13: Multi-Multi Level Models

While varying effect models are a good default fitting the heterogeneity during the fit, but how do we add multiple multi-effect models at the same time.

Prosocial Chimpanzies

Whether Chimpanzee pulls pro social option.

$$\begin{tikzcd}
T \arrow[dr] & \\
B \arrow[r] & P \\
A \arrow[ur] &
\end{tikzcd}$$

For treatment T (prosocial right no partner, left no partner, right partner, left partner), block (batch) B and actor A pulling the left leaver P. Notice because of the careful controlled setup the DAG is very clean. Now ass we expect the actor affect to be dominated by handedness we don't model it's interactions with the other parameters (In fact our DAG expects all parameters to be independent, nevertheless it might be wise to check association between T and B.)

$$\begin{aligned}
P &\sim \text{Bernoulli}(p_i) \\
logit(p_i) &= \beta_{T_i,B_i} + \alpha_{A_i} \\
\alpha_j &\sim \text{Normal}(\bar{\alpha}, \sigma_A) \\
\beta_{j,k} &\sim \text{Normal}(0, \sigma_B) \\
\sigma_j &\sim \text{Exponential}(1)
\end{aligned}$$

To some type of statistics treatments prefer fixed priors, and partial pooling is thought to be a bad choice. However obviously the treatments only influences the behaviour of the Chimpanzees, and two treatments might be similar. In essence we get better estimates through regularisation, and avoiding over and underfitting is always good.

```r
data(chimpanzees)

df <- chimpanzees

d <- list( T = (df$prosoc_left + 1) + 2*(df$condition)
         , B = df$block
         , A = df$actor
         , P = df$chose_prosoc
         , N = nrow(df))
 d$N_B <- length(unique(d$B))
 d$N_T <- length(unique(d$T))
 d$N_A <- length(unique(d$A))
```

```r
m0 <- cstan(file='../models/l13_m0.stan', data=d, chains=4, cores=4, iter=4000)
```

```
## Warning in readLines(stan_file): incomplete final line found on '../models/
## l13_m0.stan'

## Running MCMC with 4 parallel chains...
```

```
##
## Chain 1 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 1 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 1 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 2 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 2 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 3 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 3 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 4 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 4 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 1 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 2 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 4 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 1 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 2 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 4 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 4 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 1 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 1 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 2 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 1 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 2 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 2 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 3 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 4 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 4 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 2 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 3 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 4 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 2 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 3 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 4 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 2 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 1 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 2 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 2 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 3 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 1 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 1 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 2 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 4 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 1 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 2 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 2 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 3 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 4 Iteration: 1000 / 4000 [ 25%]  (Warmup)
```

```
## Chain 1 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 2 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 2 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 3 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 4 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 1 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 1 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 2 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 1 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 2 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 2 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 1 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 3 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 1 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 1 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 2 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 2 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 3 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 4 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 1 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 2 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 4 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 4 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 1 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 1 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 1 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 2 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 2 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 3 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 3 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 4 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 4 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 1 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 1 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 2 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 3 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 4 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 1 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 1 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 2 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 3 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 4 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 1 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 2 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 3 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 3 Iteration: 1500 / 4000 [ 37%]  (Warmup)
```

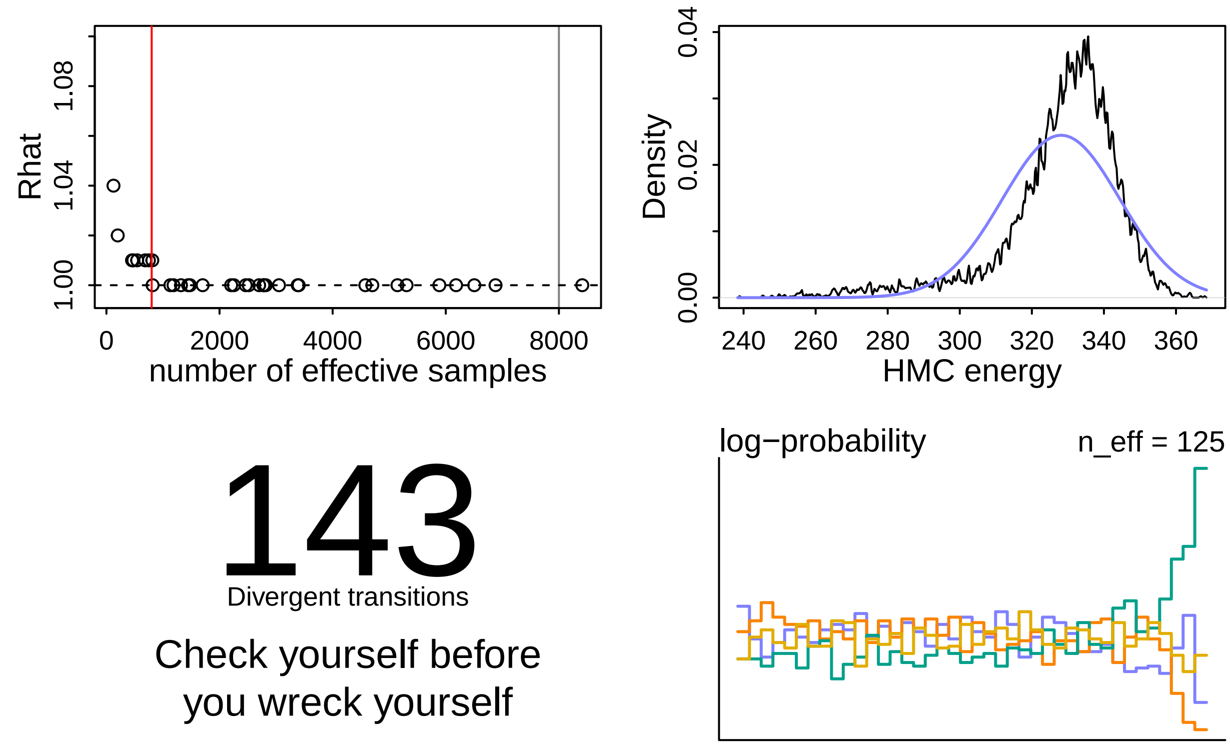
```
## Chain 4 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 1 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 1 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 2 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 3 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 1 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 2 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 3 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 4 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 4 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 1 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 1 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 2 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 3 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 4 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 1 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 1 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 2 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 4 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 1 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 2 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 2 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 3 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 3 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 4 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 1 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 1 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 2 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 3 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 4 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 1 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 2 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 3 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 4 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 4 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 1 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 1 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 2 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 4 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 1 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 2 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 2 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 3 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 4 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 1 finished in 3.1 seconds.
```

```
## Chain 2 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 3 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 4 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 2 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 3 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 4 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 2 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 3 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 4 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 4 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 2 finished in 3.4 seconds.
## Chain 4 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 3 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 3 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 4 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 4 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 3 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 4 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 3 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 4 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 4 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 4 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 3 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 4 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 4 finished in 4.1 seconds.
## Chain 3 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 3 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 3 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 3 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 3 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 3 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 3 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 3 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 3 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 3 finished in 5.3 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 4.0 seconds.
## Total execution time: 5.4 seconds.

## Warning: 143 of 8000 (2.0%) transitions ended with a divergence.
## See https://mc-stan.org/misc/warnings for details.

## Warning: 3 of 4 chains had an E-BFMI less than 0.2.
## See https://mc-stan.org/misc/warnings for details.
```

```r
precis(m0, depth=3)
```

```
##                  mean        sd        5.5%      94.5%     n_eff     Rhat4
## b[1,1]  -0.053168409 0.2576377 -0.48324981 0.34183691 6504.5576 1.0003824
## b[1,2]  -0.173000197 0.2677146 -0.65022818 0.18229902 1318.1982 1.0026694
## b[1,3]  -0.365136363 0.3298974 -0.96149064 0.04720822  475.6182 1.0077724
## b[1,4]  -0.244469950 0.2880606 -0.76843100 0.13056836  543.8224 1.0091300
## b[1,5]   0.005574469 0.2564785 -0.39960085 0.42482058 5145.4463 1.0011960
## b[1,6]  -0.056506782 0.2552911 -0.48337686 0.33453140 6182.0000 1.0015147
## b[2,1]   0.070131881 0.2582955 -0.32427852 0.50953315 5886.8229 1.0007761
## b[2,2]   0.011533784 0.2534333 -0.38753404 0.42414975 8413.3428 1.0002988
## b[2,3]  -0.049857307 0.2568787 -0.47325630 0.35034070 4701.0310 1.0001480
## b[2,4]   0.187036579 0.2847550 -0.21256935 0.69232181  816.7139 1.0023220
## b[2,5]   0.133554000 0.2637943 -0.24029296 0.59143689 2817.4839 1.0013440
## b[2,6]   0.388532266 0.3625733 -0.06621834 1.04714430  453.6173 1.0054828
## b[3,1]  -0.038178613 0.2363538 -0.42737335 0.32798761 6879.9209 1.0005921
## b[3,2]  -0.177345787 0.2947992 -0.72317842 0.22388607  685.4889 1.0070876
## b[3,3]  -0.049960413 0.2573042 -0.47514045 0.33870653 3399.6441 1.0000361
## b[3,4]   0.026690247 0.2399042 -0.34981545 0.41889718 5306.8364 0.9999751
## b[3,5]  -0.130290811 0.2879446 -0.64160411 0.27141818 2516.4853 1.0020004
## b[3,6]   0.004059572 0.2472771 -0.39915756 0.40255082 4578.0306 1.0002160
## b[4,1]  -0.209606638 0.2930478 -0.73185886 0.17377196 1186.5016 1.0040287
## b[4,2]   0.155009536 0.2590154 -0.19940857 0.60905256 1444.6834 1.0018058
## b[4,3]   0.182591986 0.2807788 -0.19709725 0.68063941 1478.3821 1.0022158
## b[4,4]   0.064364825 0.2852300 -0.34950798 0.58146170 1131.4573 1.0038599
## b[4,5]   0.061203967 0.2432815 -0.30805546 0.47215071 3382.4418 1.0007195
## b[4,6]   0.279769553 0.3191780 -0.11547805 0.85507124  751.6694 1.0063068
## a[1]     0.327360501 0.1763058  0.06635189 0.61859149 2774.8845 1.0008973
## a[2]     0.169137961 0.1884802 -0.15500423 0.43549855 2205.0275 1.0018709
## a[3]     0.351777298 0.1847209  0.08172519 0.66451195 2700.2365 1.0017084
## a[4]     0.347389281 0.1828196  0.07462489 0.66151394 2462.3504 1.0014616
## a[5]     0.332446938 0.1764953  0.06674569 0.62660269 2781.6612 1.0010361
## a[6]     0.138075391 0.1955619 -0.19459193 0.41331369 1701.7467 1.0027321
## a[7]     0.216436986 0.1765852 -0.07975649 0.47612826 3051.0000 1.0011590
## abar     0.267932483 0.1411741  0.05002541 0.48534344 2253.1226 1.0011836
## sigma_B  0.304783350 0.1525392  0.05836756 0.55766339  200.0159 1.0215621
## sigma_A  0.190173249 0.1432220  0.02417855 0.44209192  811.3596 1.0060435
```

```r
dashboard(m0)
```

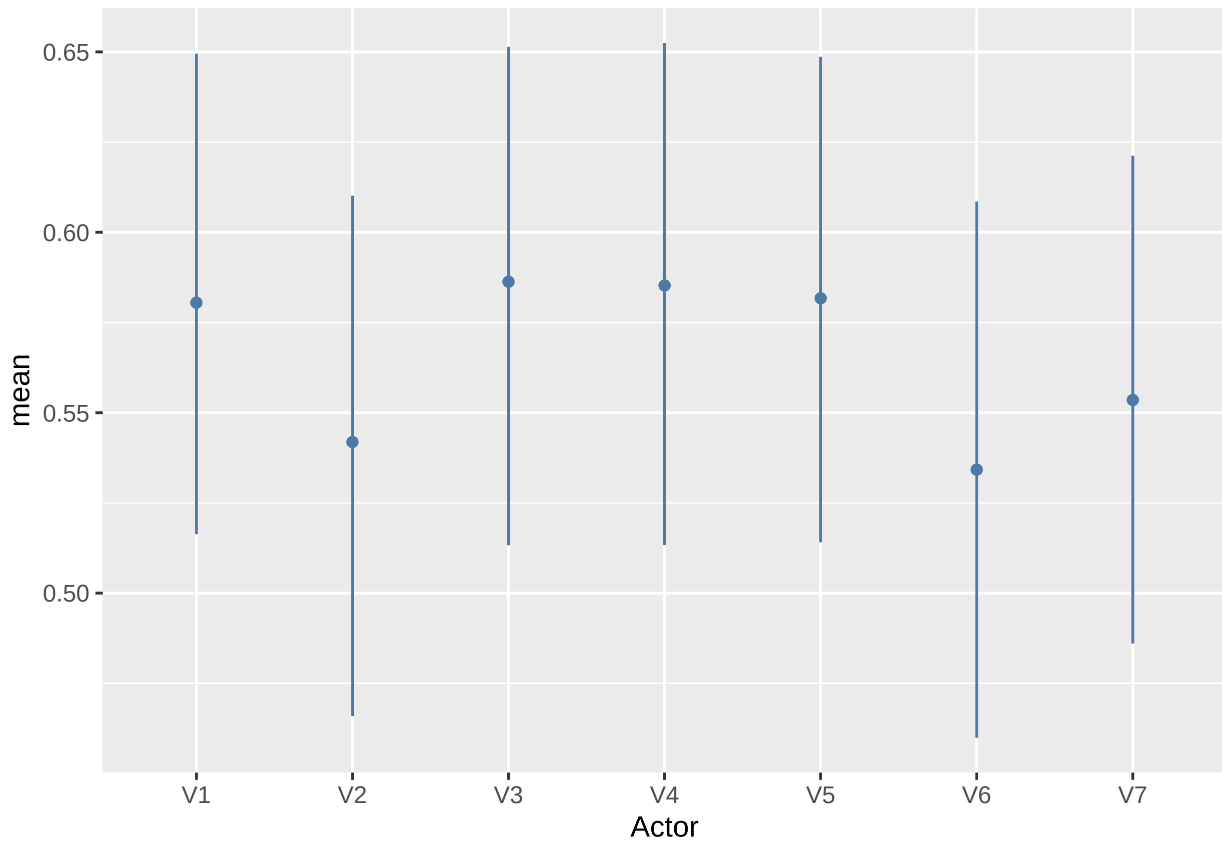


As we can see the sampling is not so effective here, our model is centered a problem **in this case**.

```
post <- extract.samples(m0)
pA <- post$a %>%
    inv_logit() %>%
    as.data.frame() %>%
    pivot_longer(everything(), names_to='Actor', names_prefix='X', values_to='p') %>%
    group_by(Actor) %>%
    summarise(mean=mean(p), hpdi=HPDI(p)) %>%
    summarise(mean=mean(mean), hpdi_lower=min(hpdi), hpdi_upper=max(hpdi))
```

```
## `summarise()` has grouped output by 'Actor'. You can override using the
## `.groups` argument.
```

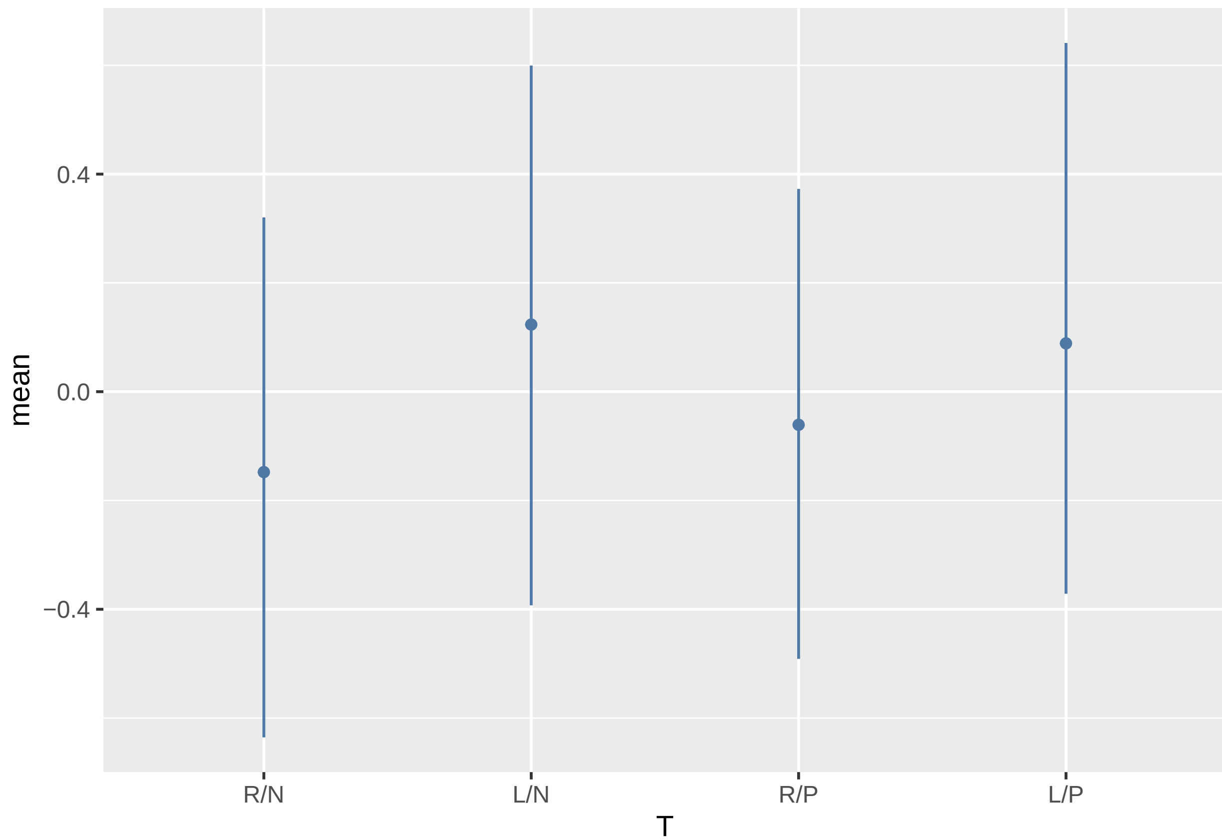
```
ggplot(pA) +
    geom_point(aes(x=Actor, y=mean, colour='')) +
    geom_segment(aes(x=Actor, xend=Actor, y=hpdi_lower, yend=hpdi_upper, colour='')) +
    theme(legend.position='none') +
    scale_color_tableau()
```



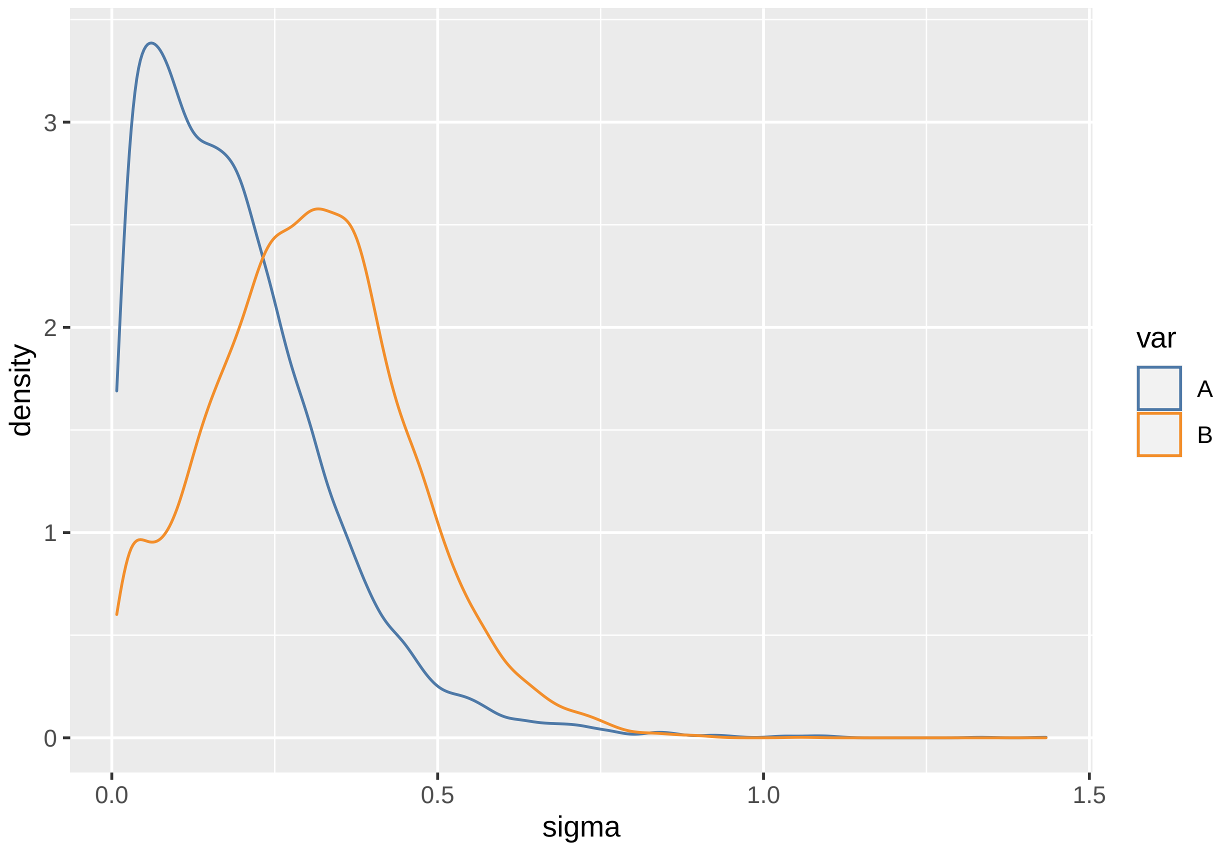

```
pB <- cbind(expand.grid(S=1:dim(post$b)[1], T=c('R/N','L/N','R/P','L/P'),B=1:dim(post$b)[3
    select(-S) %>%
    group_by(T) %>%
    summarise(mean=mean(p), hpdi=HPDI(p)) %>%
    summarise(mean=mean(mean), hpdi_lower=min(hpdi), hpdi_upper=max(hpdi))
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
ggplot(pB) +
    geom_point(aes(x=T, y=mean, colour='')) +
    geom_segment(aes(x=T, xend=T, y=hpdi_lower, yend=hpdi_upper, colour='')) +
    theme(legend.position='none')+
    scale_color_tableau()
```




```
data.frame(sigma_A=post$sigma_A, sigma_B=post$sigma_B) %>%
    pivot_longer(everything(), names_to='var', names_prefix='sigma_', values_to='sigma') %>
    ggplot() +
    geom_density(aes(x=sigma, colour=var)) +
    scale_colour_tableau()
```



Aside

Remember that we can from the posterior result generate as many imaginary samples as we want and contrast them between other conditions.

Centered Models

Now HMC can have problems with gradients of the distributions that are being sampled if the distributions depend on each other through ingranular levels of pseudo -momentum.

$$
\begin{aligned}
a &\sim \text{Normal}(0, 1) \\
\sigma &\sim \text{Normal}(b, \exp(a))
\end{aligned}
$$

is equivalent to

$$
\begin{aligned}
a &\sim \text{Normal}(0, 1) \\
\sigma &= b + z \exp(a) \\
z &\sim \text{Normal}(0, 1)
\end{aligned}
$$

but the gradients on each of the distributions is more comparable.

```r
m1 <- cstan(file='../models/l13_m1.stan', data=d, chains=4, cores=4, iter=4000)
```

```
## Warning in readLines(stan_file): incomplete final line found on '../models/
## l13_m1.stan'

## Running MCMC with 4 parallel chains...
##
## Chain 1 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 2 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 3 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 4 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 2 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 2 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 3 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 1 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 3 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 4 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 2 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 4 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 4 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 1 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 2 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 3 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 4 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 1 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 3 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 4 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 1 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 1 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 4 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 4 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 1 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 2 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 3 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 4 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 1 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 3 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 4 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 4 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 1 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 2 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 3 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 4 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 1 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 3 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 4 Iteration: 1200 / 4000 [ 30%]  (Warmup)
```

```
## Chain 1 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 3 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 4 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 4 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 1 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 1 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 2 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 3 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 4 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 2 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 3 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 1 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 3 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 4 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 1 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 1 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 2 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 3 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 3 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 4 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 1 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 1 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 2 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 3 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 3 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 4 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 3 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 4 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 1 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 2 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 3 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 4 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 4 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 3 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 4 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 2 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 3 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 3 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 4 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 2 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 3 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 4 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 2 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 3 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 3 Iteration: 2300 / 4000 [ 57%]  (Sampling)
```

```
## Chain 1 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 1 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 2 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 3 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 3 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 4 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 3 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 3 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 4 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 2 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 3 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 3 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 3 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 4 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 3 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 3 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 4 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 1 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 2 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 3 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 4 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 2 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 2 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 2 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 3 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 3 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 2 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 3 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 3 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 4 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 2 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 2 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 3 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 4 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 2 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 2 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 3 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 3 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 4 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 3 finished in 3.5 seconds.
## Chain 1 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 2 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 2 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 4 Iteration: 3200 / 4000 [ 80%]  (Sampling)
```

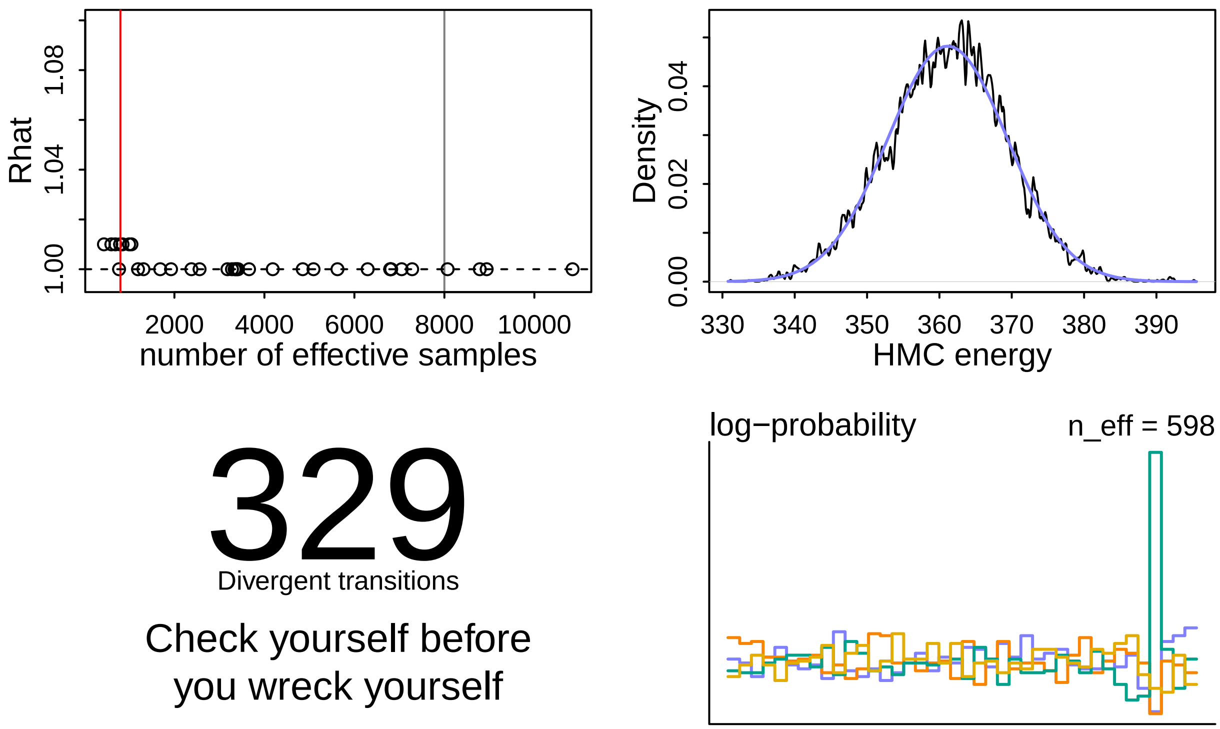
```
## Chain 2 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 4 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 2 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 2 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 4 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 1 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 2 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 4 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 2 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 2 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 2 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 2 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 4 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 2 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 2 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 4 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 2 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 4 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 2 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 4 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 2 finished in 4.5 seconds.
## Chain 4 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 4 finished in 4.6 seconds.
## Chain 1 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 1 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 1 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 1 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 1 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 1 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 1 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 1 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 1 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 1 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 1 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 1 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 1 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 1 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 1 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 1 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 1 finished in 11.8 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 6.1 seconds.
## Total execution time: 11.9 seconds.
```

```
## Warning: 329 of 8000 (4.0%) transitions ended with a divergence.
## See https://mc-stan.org/misc/warnings for details.
```

```r
precis(m1, depth=3)
```

```
##                 mean        sd        5.5%     94.5%      n_eff     Rhat4
## b[1,1]   -0.15902424 0.8518312 -1.51632000 1.2206806  8940.2663 1.0000259
## b[1,2]   -0.54553556 0.8590499 -1.84794155 0.8456289   770.0705 1.0034438
## b[1,3]   -1.03610806 0.8999953 -2.38509490 0.4820914  5091.9991 1.0014354
## b[1,4]   -0.69965015 0.8748018 -2.04634840 0.7425058  2559.4805 1.0023678
## b[1,5]    0.01280607 0.8415975 -1.33603485 1.3378758  6800.7483 1.0006300
## b[1,6]   -0.15733645 0.8284435 -1.50379730 1.1799265  6294.9631 0.9999825
## b[2,1]    0.21996944 0.8345154 -1.14235830 1.5353079  7284.5308 1.0013805
## b[2,2]    0.04363846 0.8489512 -1.32361575 1.3619109  4851.9122 1.0010892
## b[2,3]   -0.17681617 0.8542947 -1.48531495 1.1898366   791.8214 1.0078678
## b[2,4]    0.52470286 0.8787280 -0.86954318 1.9252679  2377.8275 1.0035749
## b[2,5]    0.39425511 0.8373540 -0.95801892 1.6754899  1684.4724 1.0017471
## b[2,6]    1.14164882 0.9244431 -0.41383847 2.5240749  3337.7043 1.0022618
## b[3,1]   -0.08242551 0.8280536 -1.39567275 1.1940687  1045.6477 1.0058378
## b[3,2]   -0.40487842 0.9043414 -1.82314370 1.1248600   689.6797 1.0062906
## b[3,3]   -0.12493154 0.8464987 -1.45310095 1.2143210  4185.5327 1.0016744
## b[3,4]    0.07310274 0.7985810 -1.21384115 1.3627078  8071.7082 0.9996853
## b[3,5]   -0.40941725 0.8934627 -1.72668110 1.0162927   992.9069 1.0054150
## b[3,6]    0.05537201 0.8692908 -1.33055805 1.5075146   434.2934 1.0092895
## b[4,1]   -0.60409549 0.8745598 -1.97017235 0.8110601  6794.7228 1.0003150
## b[4,2]    0.42954881 0.8179622 -0.87931750 1.7071721  6818.5350 1.0016666
## b[4,3]    0.52778433 0.8742992 -0.89311825 1.8721264  7050.9138 1.0006733
## b[4,4]    0.13469949 0.8822258 -1.27937375 1.5398637 10848.9800 1.0000563
## b[4,5]    0.19347819 0.7955307 -1.10508740 1.4419728  8787.4247 1.0007272
## b[4,6]    0.80314733 0.9015874 -0.66503568 2.2092441  5624.0772 0.9996823
## a[1]      0.32869501 0.1756876  0.06557462 0.6268449  3363.6008 1.0011682
## a[2]      0.17551704 0.1834596 -0.14138399 0.4366308  1925.4917 1.0031777
## a[3]      0.34825986 0.1785878  0.08632322 0.6611651  3391.2295 1.0008272
## a[4]      0.34821613 0.1812583  0.08235650 0.6593068  3279.4696 1.0002756
## a[5]      0.33305502 0.1742575  0.07374997 0.6215470  3658.7800 1.0007473
## a[6]      0.14471937 0.1988815 -0.20341787 0.4183190  1312.2279 1.0041284
## a[7]      0.21614655 0.1754735 -0.08242808 0.4746501  3416.0476 1.0016795
## abar      0.27040706 0.1418839  0.05010093 0.4881198  3179.6466 1.0016804
## sigma_B   0.29882604 0.1608005  0.04243689 0.5576457  1190.7296 1.0045434
## sigma_A   0.18465229 0.1465066  0.02167270 0.4463555   847.9082 1.0067408
```

```r
dashboard(m1)
```

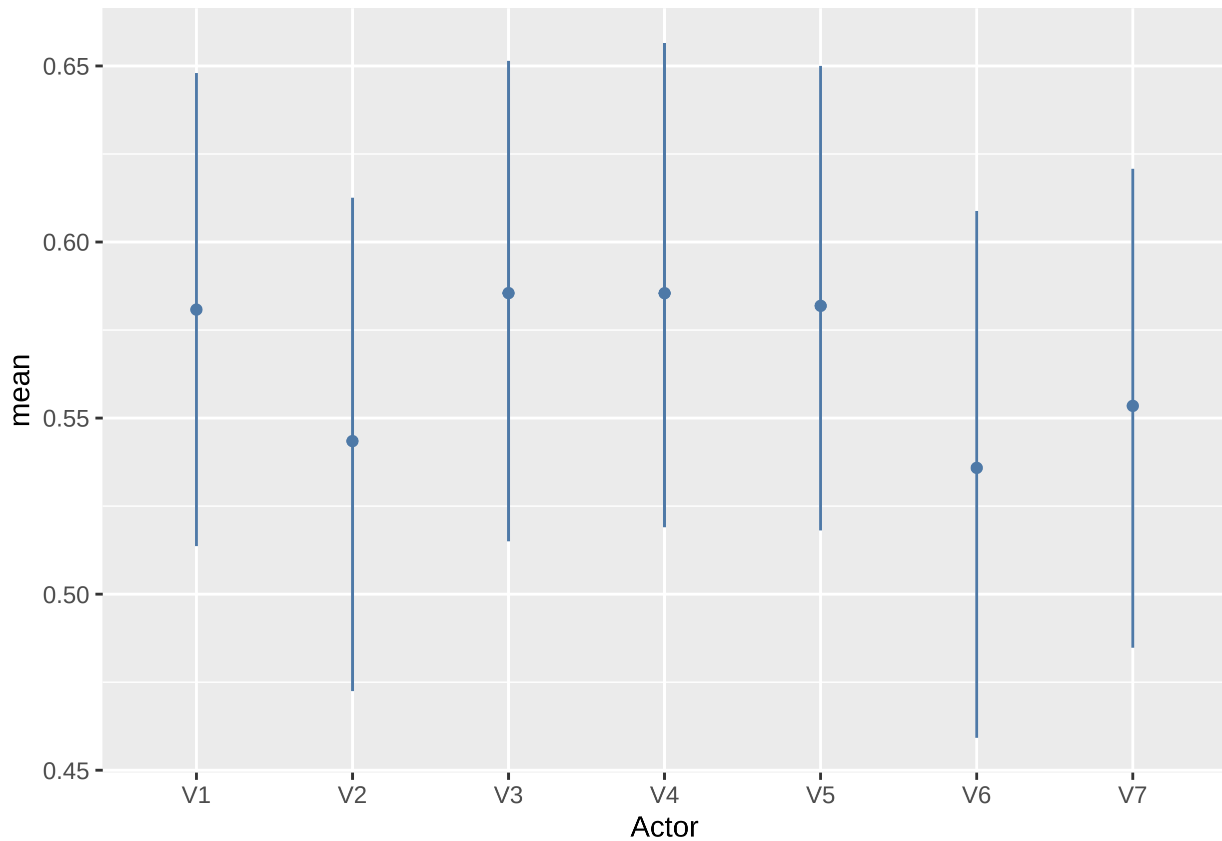


As we can see the sampling is not so effective here, our model is centered a problem **in this case**.

```
post <- extract.samples(m1)
pA <- post$a %>%
    inv_logit() %>%
    as.data.frame() %>%
    pivot_longer(everything(), names_to='Actor', names_prefix='X', values_to='p') %>%
    group_by(Actor) %>%
    summarise(mean=mean(p), hpdi=HPDI(p)) %>%
    summarise(mean=mean(mean), hpdi_lower=min(hpdi), hpdi_upper=max(hpdi))
```

```
## `summarise()` has grouped output by 'Actor'. You can override using the
## `.groups` argument.
```

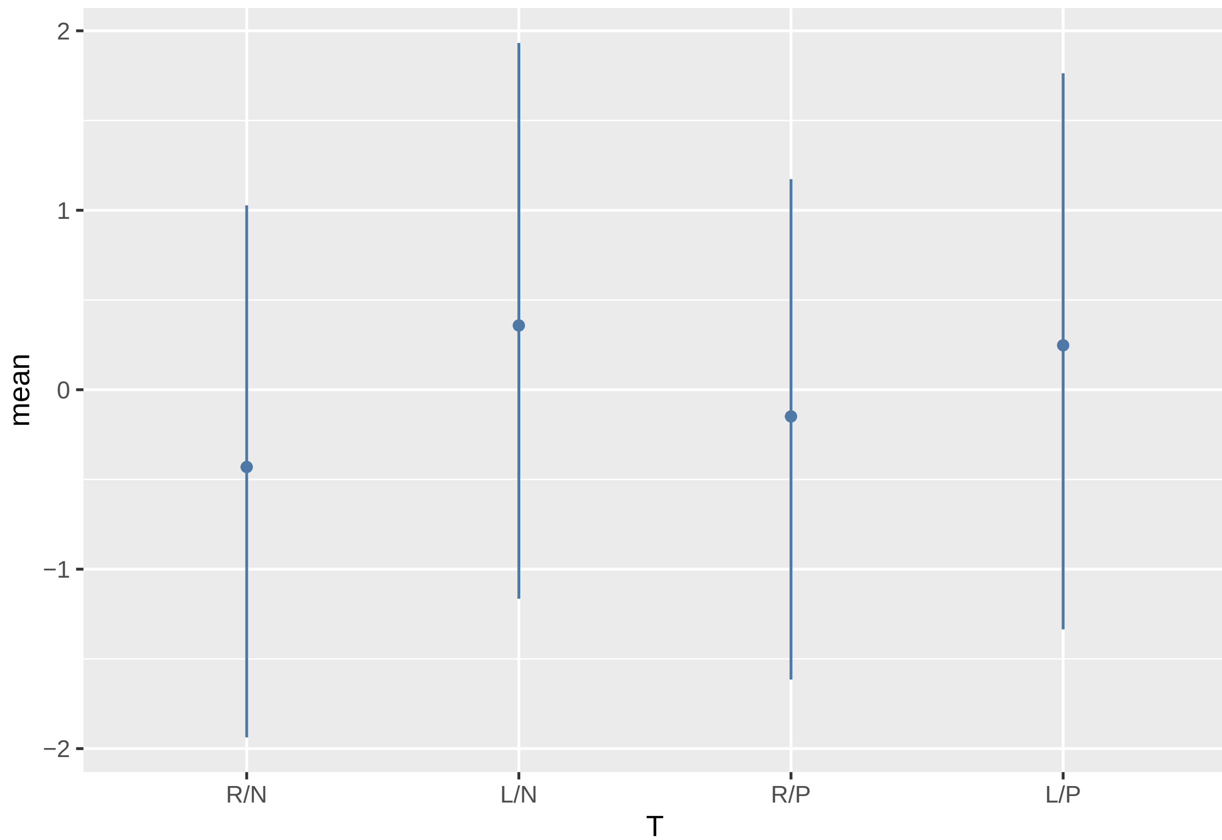
```
ggplot(pA) +
    geom_point(aes(x=Actor, y=mean, colour='')) +
    geom_segment(aes(x=Actor, xend=Actor, y=hpdi_lower, yend=hpdi_upper, colour='')) +
    theme(legend.position='none') +
    scale_color_tableau()
```



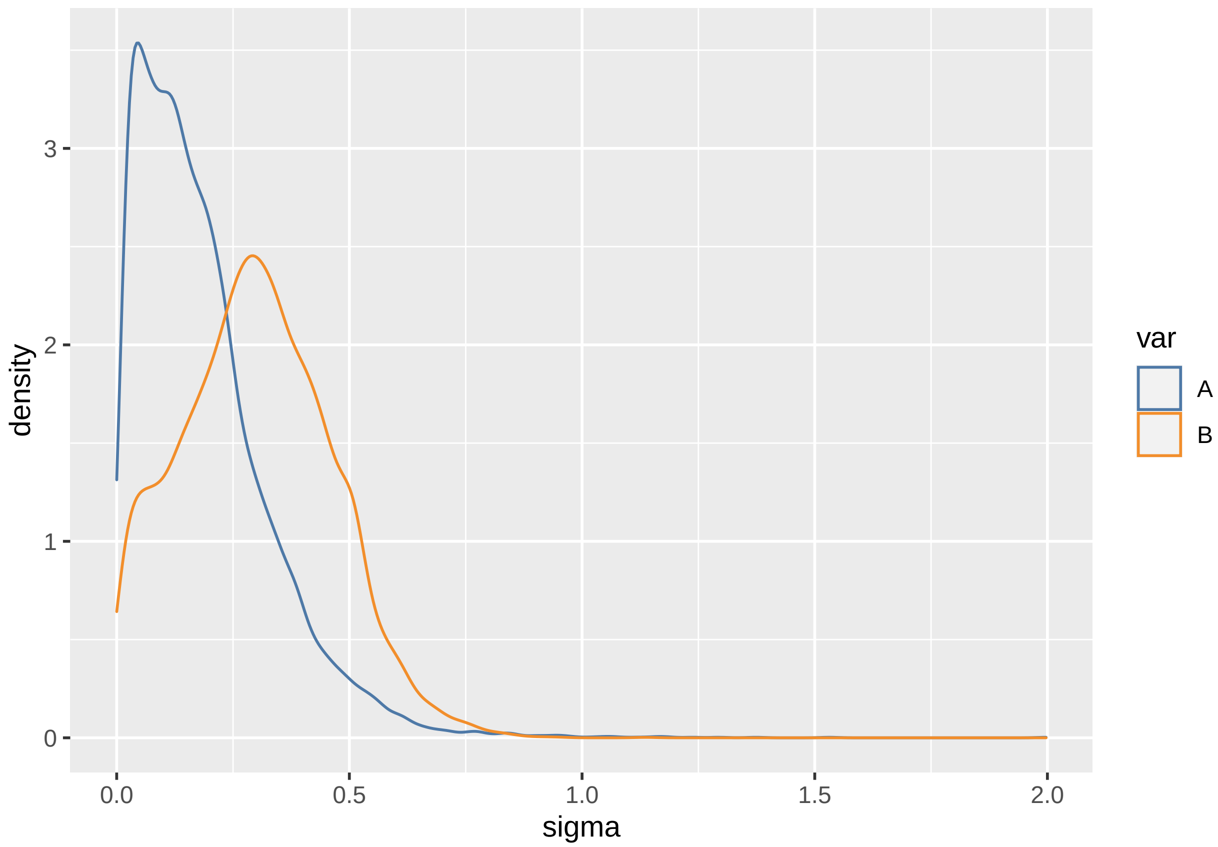

```
pB <- cbind(expand.grid(S=1:dim(post$b)[1], T=c('R/N','L/N','R/P','L/P'),B=1:dim(post$b)[3
    select(-S) %>%
    group_by(T) %>%
    summarise(mean=mean(p), hpdi=HPDI(p)) %>%
    summarise(mean=mean(mean), hpdi_lower=min(hpdi), hpdi_upper=max(hpdi))
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
ggplot(pB) +
    geom_point(aes(x=T, y=mean, colour='')) +
    geom_segment(aes(x=T, xend=T, y=hpdi_lower, yend=hpdi_upper, colour='')) +
    theme(legend.position='none')+
    scale_color_tableau()
```




```
data.frame(sigma_A=post$sigma_A, sigma_B=post$sigma_B) %>%
    pivot_longer(everything(), names_to='var', names_prefix='sigma_', values_to='sigma') %>
    ggplot() +
    geom_density(aes(x=sigma, colour=var)) +
    scale_colour_tableau()
```



Lecture 13: Correlated Varying Effects

Prosocial Chimpanzies

Same model as last chapter, whether Chimpanzee pulls pro social option.

$$\begin{tikzcd} T \arrow[dr] & \\ B \arrow[r] & P \\ A \arrow[ur] & \end{tikzcd}$$

For treatment T (prosocial right no partner, left no partner, right partner, left partner), block (batch) B and actor A pulling the left leaver P. Notice because of the careful controlled setup the DAG is very clean. Now ass we expect the actor affect to be dominated by handedness we don't model it's interactions with the other parameters (In fact our DAG expects all parameters to be independent, nevertheless it might be wise to check association between T and B.)

However now we want to try measure correlation between parameters in our multi level models, of course this requires correlation matrices. The most common prior for such a matrix is an LKJ matrix which has mean I.

$$
\begin{aligned}
P &\sim \text{Bernoulli}(p_i) \\
logit(p_i) &= \bar{\alpha}_{A_i} + \alpha_{A_i,T_i} + \bar{\beta}_{B_i} + \beta_{T_i,B_i} \\
\alpha_{j,k} &\sim \text{MVNormal}(\vec{0}, \rho_A, S_A) \\
\beta_{j,k} &\sim \text{MVNormal}(\vec{0}, \rho_B, S_B) \\
\bar{\alpha}_j &\sim \text{Normal}(0, \tau_A) \\
\bar{\beta}_j &\sim \text{Normal}(0, \tau_B) \\
\tau_j, S_k &\sim \text{Exponential}(1) \\
R_j &\sim \text{LJKCorr}(4)
\end{aligned}
$$

```r
data(chimpanzees)

df <- chimpanzees

d <- list( T = (df$prosoc_left + 1) + 2*(df$condition)
         , B = as.integer(df$block)
         , A = as.integer(df$actor)
         , P = df$pulled_left
         , N = nrow(df))
 d$N_B <- max(d$B)
 d$N_T <- max(d$T)
 d$N_A <- max(d$A)
```

```r
m0 <- cstan(file='../models/l14_m0.stan', data=d, chains=4, cores=8, threads=2, iter=4000)
```

```
## Warning in readLines(stan_file): incomplete final line found on '../models/
## l14_m0.stan'

## Running MCMC with 4 chains, at most 8 in parallel, with 2 thread(s) per chain...
##
## Chain 1 Iteration:    1 / 4000 [  0%]  (Warmup)

## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 1 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp

## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 1 but if this warning occurs often then your model may be either severely ill-cond

## Chain 1

## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 1 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp

## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 1 but if this warning occurs often then your model may be either severely ill-cond

## Chain 1
```

```
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 1 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 1 but if this warning occurs often then your model may be either severely ill-cond
## Chain 1
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 1 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 1 but if this warning occurs often then your model may be either severely ill-cond
## Chain 1
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 1 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 1 but if this warning occurs often then your model may be either severely ill-cond
## Chain 1
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 1 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 1 but if this warning occurs often then your model may be either severely ill-cond
## Chain 1
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 1 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 1 but if this warning occurs often then your model may be either severely ill-cond
## Chain 1
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 1 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 1 but if this warning occurs often then your model may be either severely ill-cond
## Chain 1
## Chain 2 Iteration:    1 / 4000 [  0%]  (Warmup)
```

```
## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 2 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp
## Chain 2 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 2 but if this warning occurs often then your model may be either severely ill-cond
## Chain 2
## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 2 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp
## Chain 2 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 2 but if this warning occurs often then your model may be either severely ill-cond
## Chain 2
## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 2 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp
## Chain 2 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 2 but if this warning occurs often then your model may be either severely ill-cond
## Chain 2
## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 2 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp
## Chain 2 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 2 but if this warning occurs often then your model may be either severely ill-cond
## Chain 2
## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 2 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp
## Chain 2 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 2 but if this warning occurs often then your model may be either severely ill-cond
## Chain 2
## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 2 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp
## Chain 2 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 2 but if this warning occurs often then your model may be either severely ill-cond
## Chain 2
## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected b
```

```
## Chain 2 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tm
## Chain 2 If this warning occurs sporadically, such as for highly constrained variable ty
## Chain 2 but if this warning occurs often then your model may be either severely ill-con
## Chain 2
## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected
## Chain 2 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tm
## Chain 2 If this warning occurs sporadically, such as for highly constrained variable ty
## Chain 2 but if this warning occurs often then your model may be either severely ill-con
## Chain 2
## Chain 3 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected
## Chain 3 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tm
## Chain 3 If this warning occurs sporadically, such as for highly constrained variable ty
## Chain 3 but if this warning occurs often then your model may be either severely ill-con
## Chain 3
## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected
## Chain 3 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tm
## Chain 3 If this warning occurs sporadically, such as for highly constrained variable ty
## Chain 3 but if this warning occurs often then your model may be either severely ill-con
## Chain 3
## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected
## Chain 3 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tm
## Chain 3 If this warning occurs sporadically, such as for highly constrained variable ty
## Chain 3 but if this warning occurs often then your model may be either severely ill-con
## Chain 3
## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected
## Chain 3 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tm
## Chain 3 If this warning occurs sporadically, such as for highly constrained variable ty
## Chain 3 but if this warning occurs often then your model may be either severely ill-con
## Chain 3
## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected
```

```
## Chain 3 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp
## Chain 3 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 3 but if this warning occurs often then your model may be either severely ill-conc
## Chain 3
## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 3 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp
## Chain 3 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 3 but if this warning occurs often then your model may be either severely ill-conc
## Chain 3
## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 3 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp
## Chain 3 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 3 but if this warning occurs often then your model may be either severely ill-conc
## Chain 3
## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 3 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp
## Chain 3 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 3 but if this warning occurs often then your model may be either severely ill-conc
## Chain 3
## Chain 4 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 4 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 4 but if this warning occurs often then your model may be either severely ill-conc
## Chain 4
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 4 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 4 but if this warning occurs often then your model may be either severely ill-conc
## Chain 4
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected b
```

```
## Chain 4 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tm
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable ty
## Chain 4 but if this warning occurs often then your model may be either severely ill-con
## Chain 4
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected
## Chain 4 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tm
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable ty
## Chain 4 but if this warning occurs often then your model may be either severely ill-con
## Chain 4
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected
## Chain 4 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tm
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable ty
## Chain 4 but if this warning occurs often then your model may be either severely ill-con
## Chain 4
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected
## Chain 4 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tm
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable ty
## Chain 4 but if this warning occurs often then your model may be either severely ill-con
## Chain 4
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected
## Chain 4 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tm
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable ty
## Chain 4 but if this warning occurs often then your model may be either severely ill-con
## Chain 4
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected
## Chain 4 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tm
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable ty
## Chain 4 but if this warning occurs often then your model may be either severely ill-con
## Chain 4
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected
## Chain 4 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tm
```

```
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 4 but if this warning occurs often then your model may be either severely ill-cond

## Chain 4

## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 4 Exception: lkj_corr_lpdf: Correlation matrix is not positive definite. (in '/tmp

## Chain 4 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 4 but if this warning occurs often then your model may be either severely ill-cond

## Chain 4

## Chain 1 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 2 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 3 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 4 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 1 Iteration:  200 / 4000 [  5%]  (Warmup)

## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 2 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat

## Chain 2 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 2 but if this warning occurs often then your model may be either severely ill-cond

## Chain 2

## Chain 3 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 2 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 1 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 2 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 1 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 2 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 3 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 4 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 3 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 2 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 4 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 2 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 3 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 4 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 2 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 1 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 4 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 2 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 3 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 4 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 3 Iteration:  700 / 4000 [ 17%]  (Warmup)
```

```
## Chain 2 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 4 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 3 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 3 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 2 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 4 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 3 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 2 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 3 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 1 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 2 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 4 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 1 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 2 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 4 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 3 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 1 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 2 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 4 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 2 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 3 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 4 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 3 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 1 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 4 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 2 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 3 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 3 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 4 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 4 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 1 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 1 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 4 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 3 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 2 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 3 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 4 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 4 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 3 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 4 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 3 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 3 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 4 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 4 Iteration: 2001 / 4000 [ 50%]  (Sampling)
```

```
## Chain 3 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 3 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 2 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 4 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 3 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 4 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 3 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 3 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 4 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 3 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 4 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 3 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 1 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 3 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 4 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 2 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 2 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 3 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 4 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 3 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 3 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 4 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 1 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 3 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 3 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 4 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 3 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 3 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 4 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 1 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 3 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 4 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 3 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 3 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 4 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 3 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 4 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 1 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 1 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 3 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 3 finished in 32.8 seconds.
## Chain 1 Iteration: 2100 / 4000 [ 52%]  (Sampling)
```

```
## Chain 4 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 1 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 4 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 1 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 4 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 1 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 2 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 1 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 4 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 1 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 4 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 1 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 1 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 4 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 4 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 1 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 1 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 4 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 4 finished in 37.8 seconds.
## Chain 1 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 1 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 1 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 1 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 1 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 1 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 1 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 2 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 1 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 1 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 1 finished in 41.8 seconds.
## Chain 2 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 2 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 2 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 2 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 2 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 2 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 2 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 2 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 2 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 2 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 2 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 2 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 2 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 2 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 2 Iteration: 3700 / 4000 [ 92%]  (Sampling)
```

```
## Chain 2 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 2 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 2 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 2 finished in 160.2 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 68.1 seconds.
## Total execution time: 160.3 seconds.

## Warning: 88 of 8000 (1.0%) transitions ended with a divergence.
## See https://mc-stan.org/misc/warnings for details.

## Warning: 1456 of 8000 (18.0%) transitions hit the maximum treedepth limit of 11.
## See https://mc-stan.org/misc/warnings for details.
```

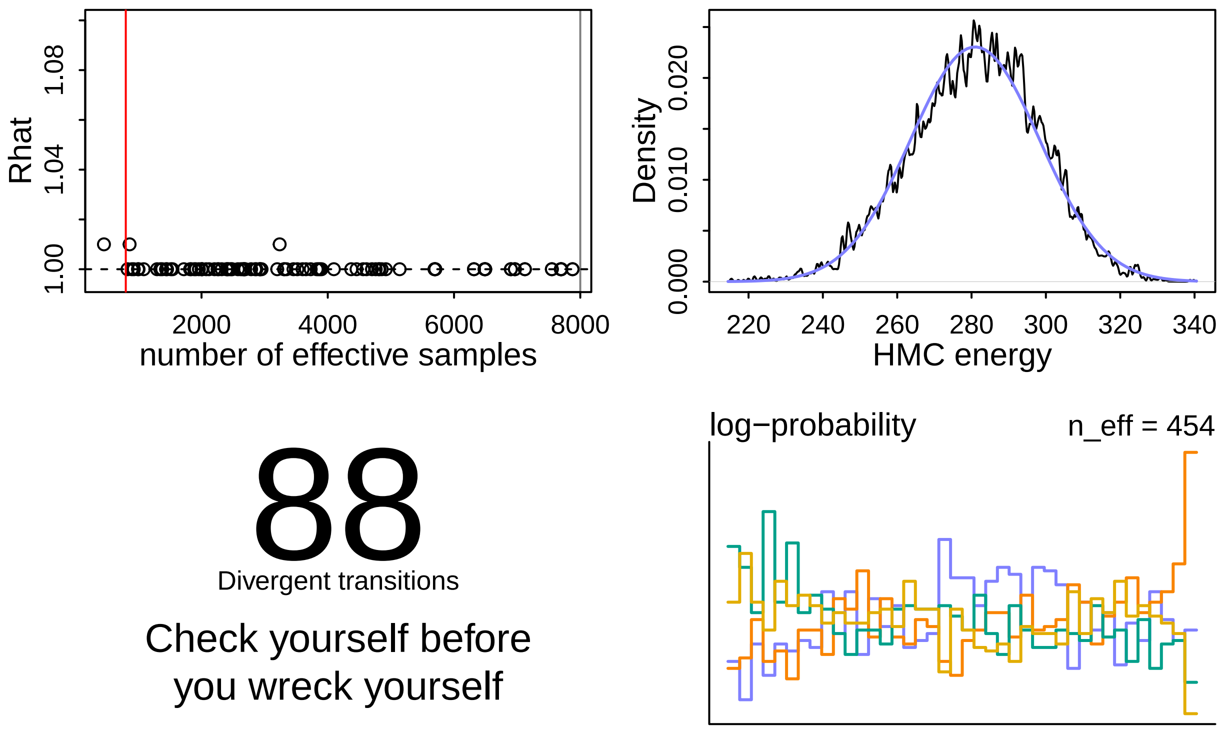
```
precis(m0, depth=3)
```

```
##                  mean        sd        5.5%      94.5%     n_eff     Rhat4
## a[1,1]     -0.1306970948 0.3592802 -0.79819376  0.33673894 2665.2705 1.0010048
## a[1,2]      0.0958506346 0.3824124 -0.46727689  0.76372330 3600.8061 1.0010544
## a[1,3]     -0.2704108749 0.4748079 -1.14022880  0.33234025 2074.2084 1.0020205
## a[1,4]      0.1282629359 0.3370317 -0.29091266  0.75315301 2408.7500 1.0001855
## a[2,1]      0.0896746632 0.5202846 -0.57956177  0.89893130 3237.7607 1.0054157
## a[2,2]      0.1061071648 0.5578078 -0.64832011  1.02723135 3824.1976 1.0004494
## a[2,3]      0.2070450567 0.7537221 -0.76634033  1.48574675 2930.6819 1.0008925
## a[2,4]      0.0784496861 0.4900270 -0.51524867  0.80782339 2676.9939 1.0024727
## a[3,1]     -0.1103925994 0.3668402 -0.77029155  0.39028536 2848.7704 1.0018918
## a[3,2]      0.4223987045 0.5153780 -0.13640690  1.39788870 1346.6105 1.0026889
## a[3,3]     -0.4415154713 0.5777833 -1.53063730  0.21652683 1453.6264 1.0018645
## a[3,4]     -0.0514916065 0.3384799 -0.62403201  0.46636099 3518.1751 1.0008967
## a[4,1]     -0.0290994390 0.3527752 -0.61726288  0.49317057 3456.8267 1.0004572
## a[4,2]      0.2258391626 0.4210713 -0.30378364  0.99765523 2205.3549 1.0018876
## a[4,3]     -0.5908783869 0.6554811 -1.80682275  0.11561236 1295.5389 1.0026218
## a[4,4]      0.0873446425 0.3443461 -0.38234122  0.70385015 2618.6527 1.0007009
## a[5,1]     -0.1175632945 0.3625227 -0.77213158  0.38069817 3309.8736 1.0007522
## a[5,2]      0.1976783710 0.4111002 -0.31695486  0.97331593 2785.7688 1.0012216
## a[5,3]     -0.2989581065 0.4872784 -1.17330650  0.30942340 1884.0115 1.0003700
## a[5,4]      0.0767527198 0.3383019 -0.38825377  0.67021554 3345.2387 1.0003675
## a[6,1]      0.2501477530 0.3976264 -0.18998427  1.00785925 2009.0680 1.0017405
## a[6,2]     -0.0880854063 0.3647858 -0.70131922  0.45669114 4854.5465 1.0007070
## a[6,3]     -0.1229814551 0.4286271 -0.86332370  0.49664732 3654.2804 1.0004910
## a[6,4]     -0.0678795766 0.3163089 -0.61690411  0.38539429 4455.9645 1.0002049
## a[7,1]     -0.1966777686 0.4407688 -1.03044495  0.32886988 2485.2781 1.0015248
## a[7,2]     -0.1575008470 0.4577868 -0.99701686  0.44467527 2922.4620 1.0001148
## a[7,3]      0.3294761770 0.6112212 -0.40979697  1.44468130 2413.1858 1.0010155
## a[7,4]      0.2737935882 0.5178340 -0.22719949  1.24740300 1513.1583 1.0010785
## b[1,1]     -0.1343723548 0.3534531 -0.76932755  0.34502053 3729.1163 0.9997868
```

```
## b[1,2]       0.1463974837 0.4015180 -0.43124085  0.85233105 3851.0190 1.0002186
## b[1,3]      -0.1447548994 0.3459607 -0.78020106  0.30466940 2443.8375 1.0030216
## b[1,4]      -0.2671349439 0.4143503 -1.04502850  0.23056452 3199.0905 1.0006658
## b[2,1]      -0.0019410875 0.3264085 -0.53821353  0.51913711 5138.4832 0.9996818
## b[2,2]      -0.0449267206 0.3866446 -0.68553300  0.56117543 4759.2279 1.0001371
## b[2,3]       0.0294037511 0.3371281 -0.49146917  0.59371302 4700.5497 1.0013261
## b[2,4]       0.2495395029 0.3853450 -0.22808966  0.94815576 2575.8561 1.0022185
## b[3,1]       0.2734644240 0.3938959 -0.17979236  1.02439365 2002.9546 1.0007901
## b[3,2]      -0.1894289122 0.4021803 -0.88539643  0.38959313 4818.5847 0.9997230
## b[3,3]      -0.0999141874 0.3492221 -0.72552453  0.38793388 2950.8398 1.0010123
## b[3,4]       0.2259754681 0.3930371 -0.25898702  0.96127654 2647.0880 1.0012477
## b[4,1]       0.0881693821 0.3402939 -0.40546948  0.68109583 4371.7655 1.0006884
## b[4,2]       0.3384882727 0.4544621 -0.25161485  1.14324520 2271.4385 1.0018919
## b[4,3]      -0.2732879837 0.4022973 -1.05374025  0.16932190 1720.2235 1.0023204
## b[4,4]       0.0926961980 0.3985799 -0.47160631  0.78940527 4561.3310 0.9998765
## b[5,1]      -0.2656887992 0.3953450 -1.02587850  0.18329374 1958.1810 1.0002991
## b[5,2]       0.2202558405 0.4244261 -0.35996087  0.98798817 2921.1204 1.0017762
## b[5,3]       0.0548249424 0.3514727 -0.47282600  0.63462551 4613.0255 1.0013180
## b[5,4]       0.0742923243 0.3438251 -0.43358208  0.66996727 4096.4228 1.0005943
## b[6,1]      -0.2159025845 0.3915651 -0.96047915  0.27043570 2255.0419 1.0006662
## b[6,2]       0.7047454126 0.6368319 -0.05301084  1.89746685 1539.7074 1.0048025
## b[6,3]      -0.2142680456 0.3986968 -0.97248375  0.25014341 1819.1462 1.0010976
## b[6,4]       0.4071664792 0.5016526 -0.15346390  1.35358055 1910.7697 1.0023441
## abar[1]     -0.3897321538 0.3769630 -0.99697441  0.19948836 2319.7156 1.0008132
## abar[2]      4.4353458863 1.2181937  2.82635990  6.54828830 4806.1549 1.0002702
## abar[3]     -0.6887592877 0.4106121 -1.34354400 -0.05727118 1837.4245 1.0024258
## abar[4]     -0.6752774864 0.4008462 -1.32752365 -0.05316691 2119.1353 1.0017963
## abar[5]     -0.3975624758 0.3866282 -1.01782980  0.21380541 2393.9846 1.0009596
## abar[6]      0.5594487232 0.3695396 -0.03385584  1.14833210 3307.8057 1.0004256
## abar[7]      2.0694457117 0.5037767  1.31127420  2.90429475 2869.4716 1.0002671
## bbar[1]     -0.1249536103 0.2240697 -0.54844975  0.12635470 2471.6082 1.0022130
## bbar[2]      0.0586367041 0.2043055 -0.21037516  0.41858993 2692.0536 1.0001371
## bbar[3]      0.0718076314 0.2103692 -0.19497513  0.45367820 2679.1814 1.0001675
## bbar[4]      0.0158164729 0.1965373 -0.28234216  0.33807578 3874.2570 0.9999348
## bbar[5]     -0.0007980383 0.2025832 -0.31388014  0.31538300 3900.3348 1.0000751
## bbar[6]      0.0999131670 0.2303262 -0.16866601  0.52340395 2004.2483 1.0012021
## tau_A        1.9812530005 0.6152555  1.18951525  3.07810360 4919.4631 1.0002078
## tau_B        0.2153363434 0.1875916  0.01781756  0.56301422  860.3191 1.0050362
## sigma_A[1]   0.4003726769 0.3259793  0.03484450  0.99917325 1002.4234 1.0048758
## sigma_A[2]   0.4665940753 0.3619681  0.04842956  1.12479145  993.1929 1.0030698
## sigma_A[3]   0.6275510916 0.4778101  0.05905883  1.50111960  895.5026 1.0035547
## sigma_A[4]   0.3633492910 0.3266255  0.03287482  0.94370021  828.0213 1.0034658
## sigma_B[1]   0.4129608736 0.3175253  0.04108565  0.99368125  926.8505 1.0018005
## sigma_B[2]   0.5740018534 0.3938290  0.08141202  1.28788685 1437.9338 1.0047944
## sigma_B[3]   0.3878283353 0.3201654  0.04271684  0.97903724 1084.0683 1.0031749
## sigma_B[4]   0.4750573293 0.3633089  0.05911970  1.12876585 1377.0207 1.0026627
```

```
## Rho_A[1,1]  1.0000000000 0.0000000  1.00000000  1.00000000       NaN       NaN
## Rho_A[1,2]  0.0100573480 0.3022107 -0.48635300  0.49163896 7122.5844 0.9997399
## Rho_A[1,3]  0.0490126074 0.3085747 -0.45474587  0.53377456 6315.6937 0.9997681
## Rho_A[1,4]  0.0031158522 0.3046241 -0.48383511  0.49643025 7549.6826 1.0002500
## Rho_A[2,1]  0.0100573480 0.3022107 -0.48635300  0.49163896 7122.5844 0.9997399
## Rho_A[2,2]  1.0000000000 0.0000000  1.00000000  1.00000000       NaN       NaN
## Rho_A[2,3] -0.0459822917 0.2999337 -0.52759542  0.44745940 6484.5433 1.0000924
## Rho_A[2,4]  0.0265080278 0.3040605 -0.46515540  0.50324892 6900.9784 1.0000848
## Rho_A[3,1]  0.0490126074 0.3085747 -0.45474587  0.53377456 6315.6937 0.9997681
## Rho_A[3,2] -0.0459822917 0.2999337 -0.52759542  0.44745940 6484.5433 1.0000924
## Rho_A[3,3]  1.0000000000 0.0000000  1.00000000  1.00000000       NaN       NaN
## Rho_A[3,4]  0.0373283498 0.2991783 -0.44965580  0.51272079 5703.3328 0.9999835
## Rho_A[4,1]  0.0031158522 0.3046241 -0.48383511  0.49643025 7549.6826 1.0002500
## Rho_A[4,2]  0.0265080278 0.3040605 -0.46515540  0.50324892 6900.9784 1.0000848
## Rho_A[4,3]  0.0373283498 0.2991783 -0.44965580  0.51272079 5703.3328 0.9999835
## Rho_A[4,4]  1.0000000000 0.0000000  1.00000000  1.00000000       NaN       NaN
## Rho_B[1,1]  1.0000000000 0.0000000  1.00000000  1.00000000       NaN       NaN
## Rho_B[1,2] -0.0548571909 0.2982688 -0.52777483  0.43100084 7877.7419 1.0003263
## Rho_B[1,3]  0.0064141111 0.3014484 -0.48278242  0.48591733 7691.6024 0.9996502
## Rho_B[1,4]  0.0203771762 0.3026771 -0.46774169  0.51031198 7699.9297 0.9997309
## Rho_B[2,1] -0.0548571909 0.2982688 -0.52777483  0.43100084 7877.7419 1.0003263
## Rho_B[2,2]  1.0000000000 0.0000000  1.00000000  1.00000000       NaN       NaN
## Rho_B[2,3] -0.0511688153 0.3010168 -0.53038310  0.43663619 6963.9519 1.0000535
## Rho_B[2,4]  0.0692444393 0.3050420 -0.43254090  0.54553255 5691.6048 1.0004402
## Rho_B[3,1]  0.0064141111 0.3014484 -0.48278242  0.48591733 7691.6024 0.9996502
## Rho_B[3,2] -0.0511688153 0.3010168 -0.53038310  0.43663619 6963.9519 1.0000535
## Rho_B[3,3]  1.0000000000 0.0000000  1.00000000  1.00000000       NaN       NaN
## Rho_B[3,4] -0.0084639279 0.3008270 -0.49013482  0.47651653 6494.6971 1.0003209
## Rho_B[4,1]  0.0203771762 0.3026771 -0.46774169  0.51031198 7699.9297 0.9997309
## Rho_B[4,2]  0.0692444393 0.3050420 -0.43254090  0.54553255 5691.6048 1.0004402
## Rho_B[4,3] -0.0084639279 0.3008270 -0.49013482  0.47651653 6494.6971 1.0003209
## Rho_B[4,4]  1.0000000000 0.0000000  1.00000000  1.00000000       NaN       NaN
```

```
dashboard(m0)
```



As we can see the sampling is not so effective here, our model is centered a problem **in this case**. But how does one do this with these matrix terms? Can decompose out the Cholesky Factors L_A.

$$\alpha = (\text{diag}(S_A)L_A Z_{T,A})^T$$

Giving us a equivalent non-centred model

$$
\begin{aligned}
P &\sim \text{Bernoulli}(p_i) \\
logit(p_i) &= \bar{\alpha}_{A_i} + \alpha_{A_i,T_i} + \bar{\beta}_{B_i} + \beta_{T_i,B_i} \\
\alpha_{j,k} &\sim (\text{diag}(S_A)L_A Z_{T,A})^T \\
\beta_{j,k} &\sim (\text{diag}(S_B)L_B Z_{T,B})^T \\
Z_{T,A}, Z_{T,B} &\sim \text{Normal}(0,1) \\
z_{\bar{\alpha},j}, z_{\bar{\beta},j} &\sim \text{Normal}(0,1) \\
\bar{\alpha}_j &= z_{\bar{\alpha}}\tau_A \\
\bar{\beta}_j &= z_{\bar{\beta}}\tau_B \\
\vec{\tau}, \vec{S} &\sim \text{Exponential}(1) \\
\vec{R} &\sim \text{LJKCorr}(4)
\end{aligned}
$$

```
m1 <- cstan(file='../models/l14_m1.stan', data=d, chains=4, cores=8, threads=2, iter=4000)
```

```
## Warning in readLines(stan_file): incomplete final line found on '../models/
## l14_m1.stan'

## Running MCMC with 4 chains, at most 8 in parallel, with 2 thread(s) per chain...
```

```
##
## Chain 1 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 2 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 3 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 4 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 1 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 2 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 3 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 4 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 2 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 1 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 3 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 4 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 2 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 2 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 3 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 4 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 1 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 3 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 2 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 4 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 1 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 3 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 4 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 1 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 2 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 3 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 4 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 1 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 2 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 2 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 3 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 4 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 1 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 4 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 1 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 2 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 3 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 2 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 3 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 4 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 1 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 3 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 2 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 4 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 1 Iteration: 1000 / 4000 [ 25%]  (Warmup)
```

```
## Chain 3 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 1 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 2 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 4 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 3 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 1 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 2 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 4 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 3 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 2 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 4 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 1 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 3 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 2 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 4 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 1 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 2 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 3 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 1 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 3 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 4 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 2 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 4 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 1 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 2 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 3 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 1 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 4 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 2 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 3 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 2 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 2 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 2 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 3 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 4 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 1 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 1 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 2 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 3 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 3 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 1 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 2 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 4 Iteration: 2000 / 4000 [ 50%]  (Warmup)
```

```
## Chain 4 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 1 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 3 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 2 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 3 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 4 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 1 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 2 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 3 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 4 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 1 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 2 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 3 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 4 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 1 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 2 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 3 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 4 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 1 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 2 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 3 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 4 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 2 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 1 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 3 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 4 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 1 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 2 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 3 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 4 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 1 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 2 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 3 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 4 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 1 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 2 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 4 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 3 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 1 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 2 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 3 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 4 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 1 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 3 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 2 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 4 Iteration: 3100 / 4000 [ 77%]  (Sampling)
```

```
## Chain 1 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 3 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 2 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 4 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 1 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 3 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 1 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 2 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 3 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 4 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 1 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 3 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 1 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 2 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 3 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 4 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 1 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 2 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 3 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 4 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 1 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 4 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 1 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 2 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 3 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 4 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 1 finished in 7.0 seconds.
## Chain 4 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 2 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 3 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 2 finished in 7.1 seconds.
## Chain 3 finished in 7.1 seconds.
## Chain 4 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 4 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 4 finished in 7.3 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 7.1 seconds.
## Total execution time: 7.4 seconds.
```

```
precis(m1, depth=3)
```

```
##                     mean         sd        5.5%        94.5%      n_eff
## zA[1,1]      -0.230549060 0.86389901 -1.60578550  1.155373200  7313.194
## zA[1,2]       0.144411383 0.98525230 -1.47291080  1.707566650 10523.950
## zA[1,3]      -0.224283379 0.90077875 -1.63776825  1.249705400  7022.608
```

```
## zA[1,4]      -0.063733899 0.88038233 -1.47339755  1.339009050  6256.785
## zA[1,5]      -0.237223072 0.86428515 -1.58340310  1.168143150  7809.420
## zA[1,6]       0.469138701 0.87886100 -0.96464223  1.830734050  7832.086
## zA[1,7]      -0.354243608 0.94297363 -1.81752665  1.202441350  8408.982
## zA[2,1]       0.183571805 0.84045014 -1.20517540  1.497258300  7326.560
## zA[2,2]       0.118511764 0.97353227 -1.43526670  1.664371050 11333.307
## zA[2,3]       0.727001680 0.91880794 -0.82049353  2.107368250  5406.859
## zA[2,4]       0.388498574 0.87753457 -1.08071575  1.736502550  6007.912
## zA[2,5]       0.344572819 0.86918392 -1.06921160  1.712078850  6505.592
## zA[2,6]      -0.187830663 0.83936541 -1.52232100  1.137863350  8724.585
## zA[2,7]      -0.226630676 0.89353847 -1.64334700  1.206984750  8735.527
## zA[3,1]      -0.324706234 0.83894454 -1.64009440  1.039819600  8198.958
## zA[3,2]       0.201201502 0.98121138 -1.35777455  1.733489650 10815.365
## zA[3,3]      -0.522781571 0.84610460 -1.84061925  0.864505420  7160.222
## zA[3,4]      -0.758346434 0.88182377 -2.10318355  0.730371980  6595.244
## zA[3,5]      -0.365738819 0.83375632 -1.65018380  0.970087635  7790.242
## zA[3,6]      -0.188277696 0.82181384 -1.43623385  1.149219100  8410.509
## zA[3,7]       0.432555106 0.90168461 -1.03571590  1.837951900  9235.448
## zA[4,1]       0.267248645 0.93212622 -1.26412905  1.725826600  9815.218
## zA[4,2]       0.088179865 0.99688915 -1.52463100  1.662818550 12702.093
## zA[4,3]      -0.111765082 0.92710840 -1.60669705  1.377891800  8176.112
## zA[4,4]       0.178395516 0.94705730 -1.37118805  1.677604200  9090.558
## zA[4,5]       0.163345394 0.90564641 -1.28767170  1.607121900  8989.247
## zA[4,6]      -0.110421493 0.91083190 -1.57679825  1.358426500 10313.217
## zA[4,7]       0.451300060 1.00649602 -1.18292915  2.047803200  9195.431
## zB[1,1]      -0.274364502 0.84980990 -1.61666770  1.101447500  7344.669
## zB[1,2]       0.006397040 0.82627690 -1.28713385  1.315670500  7724.163
## zB[1,3]       0.538118918 0.85340241 -0.88818253  1.858719700  6356.089
## zB[1,4]       0.184388584 0.84519293 -1.15308980  1.546831600  7963.462
## zB[1,5]      -0.538967375 0.84533576 -1.86145630  0.849151670  7032.440
## zB[1,6]      -0.423096808 0.87289564 -1.77241535  1.018611150  6328.728
## zB[2,1]       0.166424618 0.80300265 -1.13966125  1.395844300  7247.487
## zB[2,2]      -0.127489835 0.78539575 -1.38677895  1.110578300  8147.862
## zB[2,3]      -0.317538005 0.79369858 -1.59439840  0.954322040  8185.382
## zB[2,4]       0.486734871 0.82831797 -0.88734353  1.756795750  6412.097
## zB[2,5]       0.272269974 0.80034647 -1.04071310  1.532732050  7101.410
## zB[2,6]       1.023573524 0.89753412 -0.53787786  2.372737150  4867.895
## zB[3,1]      -0.255153191 0.92826414 -1.69265400  1.271834900  8949.075
## zB[3,2]       0.066331797 0.89736951 -1.35997705  1.512353200 10297.420
## zB[3,3]      -0.180083255 0.91254087 -1.61523605  1.309639150  8919.530
## zB[3,4]      -0.525177629 0.90646845 -1.93488330  0.978957430  8649.540
## zB[3,5]       0.157310450 0.89809569 -1.29099960  1.592864300  9932.869
## zB[3,6]      -0.344777460 0.93039967 -1.78642165  1.195203300  8588.706
## zB[4,1]      -0.475481087 0.89723821 -1.89681015  0.973369440  8873.923
## zB[4,2]       0.407623011 0.85484903 -0.99629091  1.739970400  8857.700
## zB[4,3]       0.345206477 0.89117859 -1.10725770  1.715769350  9607.307
```

```
## zB[4,4]        0.057282250 0.86720101 -1.34842690  1.426635450 11176.284
## zB[4,5]        0.080123386 0.84987098 -1.27977535  1.429107600 10127.549
## zB[4,6]        0.573916787 0.92545534 -0.94640041  2.005835200  7890.746
## zAbar[1]      -0.220308947 0.23141492 -0.61542748  0.103901525  3139.715
## zAbar[2]       2.343820963 0.60638613  1.43877955  3.369653700  4551.676
## zAbar[3]      -0.372891488 0.25906316 -0.82221255 -0.018633066  2395.416
## zAbar[4]      -0.364251451 0.25292022 -0.79963245 -0.009852031  2444.797
## zAbar[5]      -0.216701192 0.22992766 -0.59581562  0.110291325  2831.836
## zAbar[6]       0.308291866 0.23258095 -0.02231199  0.705758165  3895.083
## zAbar[7]       1.120987339 0.38852984  0.57015342  1.798127950  3305.108
## zBbar[1]      -0.484330854 0.92225428 -1.91284735  1.052350050  7230.234
## zBbar[2]       0.192806390 0.88281374 -1.21967815  1.572742450  9258.306
## zBbar[3]       0.265380738 0.90382489 -1.20857025  1.674123750  7598.600
## zBbar[4]       0.038999380 0.87234859 -1.35448355  1.417580250  8580.919
## zBbar[5]      -0.014634374 0.90201165 -1.44805990  1.437558700  9387.512
## zBbar[6]       0.344801873 0.90624968 -1.12810105  1.782349800  8347.558
## tau_A          1.992976779 0.65237264  1.17746555  3.184938500  2780.463
## tau_B          0.224784607 0.19638577  0.01665869  0.575483665  3398.946
## sigma_A[1]     0.383870969 0.32645356  0.02646431  0.971776715  4087.828
## sigma_A[2]     0.463005168 0.36556379  0.03719006  1.109568850  3217.462
## sigma_A[3]     0.629787045 0.47203223  0.05647112  1.484442600  3092.160
## sigma_A[4]     0.371701644 0.33448857  0.02619260  0.983602460  4474.334
## sigma_B[1]     0.404854972 0.31410142  0.03806197  0.977596825  4540.573
## sigma_B[2]     0.551238247 0.40399743  0.05426105  1.271601600  3216.281
## sigma_B[3]     0.372178241 0.32076344  0.02481965  0.956818050  4584.654
## sigma_B[4]     0.468435618 0.36371207  0.03983842  1.123229150  3227.250
## L_Rho_A[1,1]   1.000000000 0.00000000  1.00000000  1.000000000       NaN
## L_Rho_A[1,2]   0.000000000 0.00000000  0.00000000  0.000000000       NaN
## L_Rho_A[1,3]   0.000000000 0.00000000  0.00000000  0.000000000       NaN
## L_Rho_A[1,4]   0.000000000 0.00000000  0.00000000  0.000000000       NaN
## L_Rho_A[2,1]   0.008494181 0.29831040 -0.47219382  0.491518995  8747.145
## L_Rho_A[2,2]   0.952274426 0.06421693  0.82632235  0.999771055  4379.136
## L_Rho_A[2,3]   0.000000000 0.00000000  0.00000000  0.000000000       NaN
## L_Rho_A[2,4]   0.000000000 0.00000000  0.00000000  0.000000000       NaN
## L_Rho_A[3,1]   0.043365503 0.30182758 -0.45205797  0.518570335  7514.347
## L_Rho_A[3,2]  -0.054367538 0.29929284 -0.52873614  0.436498540  7497.722
## L_Rho_A[3,3]   0.897878639 0.09124191  0.72153557  0.993381540  4336.803
## L_Rho_A[3,4]   0.000000000 0.00000000  0.00000000  0.000000000       NaN
## L_Rho_A[4,1]   0.004980725 0.30234721 -0.48341318  0.481705465 10722.785
## L_Rho_A[4,2]   0.032689731 0.30504537 -0.45863236  0.519894600  9157.701
## L_Rho_A[4,3]   0.038423170 0.29738421 -0.43605829  0.510332960  8992.118
## L_Rho_A[4,4]   0.844059927 0.11011281  0.63693796  0.977161660  3792.021
## L_Rho_B[1,1]   1.000000000 0.00000000  1.00000000  1.000000000       NaN
## L_Rho_B[1,2]   0.000000000 0.00000000  0.00000000  0.000000000       NaN
## L_Rho_B[1,3]   0.000000000 0.00000000  0.00000000  0.000000000       NaN
## L_Rho_B[1,4]   0.000000000 0.00000000  0.00000000  0.000000000       NaN
```

```
## L_Rho_B[2,1] -0.053082664 0.29859316 -0.52635807  0.440238935  6974.250
## L_Rho_B[2,2]  0.950626135 0.06592421  0.81940782  0.999747055  4197.538
## L_Rho_B[2,3]  0.000000000 0.00000000  0.00000000  0.000000000       NaN
## L_Rho_B[2,4]  0.000000000 0.00000000  0.00000000  0.000000000       NaN
## L_Rho_B[3,1]  0.007164639 0.29808511 -0.46744963  0.478663615 10006.219
## L_Rho_B[3,2] -0.047340858 0.29729408 -0.51393126  0.440526135  9298.941
## L_Rho_B[3,3]  0.901530839 0.08794474  0.73228587  0.993386265  3964.005
## L_Rho_B[3,4]  0.000000000 0.00000000  0.00000000  0.000000000       NaN
## L_Rho_B[4,1]  0.027543456 0.29385578 -0.44019707  0.495376280  8792.906
## L_Rho_B[4,2]  0.059114530 0.29874635 -0.42553312  0.532322445  8030.345
## L_Rho_B[4,3] -0.007408000 0.29707557 -0.47634931  0.471980330  8631.274
## L_Rho_B[4,4]  0.848364431 0.11022589  0.64070248  0.978626715  4201.687
## a[1,1]       -0.113615014 0.34665326 -0.75330922  0.354058435  5738.349
## a[1,2]        0.114392659 0.37573638 -0.41121950  0.766470265  5872.901
## a[1,3]       -0.275065735 0.47022979 -1.14640540  0.320131815  4885.273
## a[1,4]        0.143046720 0.35898354 -0.30206868  0.788877300  5464.061
## a[2,1]        0.090229540 0.48640580 -0.53663951  0.874763640  8179.679
## a[2,2]        0.098323318 0.55660810 -0.63560965  1.037727350  8160.771
## a[2,3]        0.205131598 0.73259366 -0.72490390  1.469510800  6646.762
## a[2,4]        0.082435890 0.49835082 -0.54683969  0.888139050  6933.785
## a[3,1]       -0.113336190 0.36300005 -0.75647202  0.384112070  5258.230
## a[3,2]        0.427154981 0.51589396 -0.13273705  1.419012200  3501.956
## a[3,3]       -0.455900994 0.57940618 -1.51451110  0.219259435  3723.359
## a[3,4]       -0.047879640 0.34936033 -0.63456914  0.470975410  6450.369
## a[4,1]       -0.033281646 0.34858131 -0.62625547  0.488531360  5829.278
## a[4,2]        0.228438071 0.42546719 -0.29237998  1.032440450  4252.425
## a[4,3]       -0.604238410 0.65216475 -1.80368930  0.122414170  3386.422
## a[4,4]        0.087603774 0.35501704 -0.41353792  0.710216815  6234.686
## a[5,1]       -0.119457126 0.35852867 -0.75890938  0.350803635  6478.199
## a[5,2]        0.200192912 0.40296381 -0.31557320  0.946967490  4636.510
## a[5,3]       -0.309726469 0.48627608 -1.21052605  0.298311210  4551.462
## a[5,4]        0.079445222 0.33255416 -0.37994247  0.675489470  6093.881
## a[6,1]        0.234842857 0.39735648 -0.20591798  1.012823850  5502.306
## a[6,2]       -0.092283344 0.37050967 -0.74223682  0.450070115  7946.492
## a[6,3]       -0.124379284 0.42485304 -0.87074663  0.499131315  6429.758
## a[6,4]       -0.061339113 0.33554058 -0.63168651  0.413739245  7044.431
## a[7,1]       -0.185898266 0.42431243 -0.98236339  0.329963855  6674.669
## a[7,2]       -0.137468841 0.43485218 -0.91035955  0.451449445  7469.387
## a[7,3]        0.343584377 0.61778291 -0.38029973  1.470646400  6413.301
## a[7,4]        0.285327289 0.54977096 -0.22488603  1.293534350  5097.088
## b[1,1]       -0.131811531 0.34580506 -0.75362006  0.333771605  6911.782
## b[1,2]        0.130636806 0.39739324 -0.43641032  0.834516750  6152.495
## b[1,3]       -0.137576489 0.35001397 -0.78670317  0.307754985  5869.169
## b[1,4]       -0.273902140 0.42485012 -1.06507100  0.221371675  5900.362
## b[2,1]       -0.002043586 0.32673732 -0.53962970  0.523231590  9610.123
## b[2,2]       -0.060796672 0.37558724 -0.69989309  0.524913300  8025.411
```

```
## b[2,3]         0.021205300 0.32938105 -0.49217630  0.540349870  7894.072
## b[2,4]         0.238064537 0.37817901 -0.22236386  0.952682980  4392.101
## b[3,1]         0.260190240 0.38659310 -0.18573824  0.991310295  6107.087
## b[3,2]        -0.194083477 0.39901614 -0.90246578  0.375129960  8089.920
## b[3,3]        -0.093937960 0.34536321 -0.73230466  0.375126820  7458.334
## b[3,4]         0.222502998 0.39396497 -0.26609748  0.944312440  5379.520
## b[4,1]         0.085318891 0.33374316 -0.39779104  0.678125740  8771.140
## b[4,2]         0.312898135 0.44540739 -0.24354056  1.147954300  4833.510
## b[4,3]        -0.264403372 0.39590766 -1.02275310  0.158394110  4621.579
## b[4,4]         0.079389722 0.38842334 -0.48218546  0.759053720  7438.917
## b[5,1]        -0.267246402 0.39316557 -1.01008125  0.177531370  6332.901
## b[5,2]         0.202259506 0.40592162 -0.34984755  0.938125045  5739.585
## b[5,3]         0.056092713 0.33042813 -0.43128466  0.620724855  8338.765
## b[5,4]         0.072058075 0.34667481 -0.43177926  0.677859420  7185.306
## b[6,1]        -0.213681313 0.38754098 -0.93674798  0.266097130  5746.999
## b[6,2]         0.668955356 0.62355517 -0.05828448  1.833274350  3242.073
## b[6,3]        -0.206800989 0.39023454 -0.94453690  0.245665065  5496.841
## b[6,4]         0.399217976 0.50862461 -0.16314747  1.384048750  3764.799
## abar[1]       -0.395975587 0.38053222 -1.01009880  0.195725160  3811.242
## abar[2]        4.462260458 1.29014365  2.82161745  6.700242550  5574.730
## abar[3]       -0.678666016 0.41207804 -1.34400220 -0.036078925  2978.104
## abar[4]       -0.663346798 0.40818759 -1.31890850 -0.019009193  3281.264
## abar[5]       -0.389674504 0.38237116 -1.00083040  0.215037860  3619.130
## abar[6]        0.566855599 0.37921534 -0.04203303  1.167125550  5082.953
## abar[7]        2.063650807 0.49790375  1.29291325  2.872963750  5776.067
## bbar[1]       -0.131161300 0.23248883 -0.56582415  0.143327395  5163.656
## bbar[2]        0.060277945 0.21145202 -0.22098357  0.436149555  5508.662
## bbar[3]        0.076249198 0.22083992 -0.20847460  0.471781740  4862.612
## bbar[4]        0.018255366 0.20569748 -0.28046991  0.344178890  6668.104
## bbar[5]       -0.003151246 0.20565507 -0.33167775  0.313146440  7130.042
## bbar[6]        0.100526647 0.23132562 -0.17612501  0.514218745  4949.653
## p[1]           0.325406581 0.10378694  0.16644708  0.498189625  8557.911
## p[2]           0.325406581 0.10378694  0.16644708  0.498189625  8557.911
## p[3]           0.433814638 0.11768975  0.25102506  0.630219655 10176.261
## p[4]           0.325406581 0.10378694  0.16644708  0.498189625  8557.911
## p[5]           0.433814638 0.11768975  0.25102506  0.630219655 10176.261
## p[6]           0.433814638 0.11768975  0.25102506  0.630219655 10176.261
## p[7]           0.433999187 0.11703034  0.25000170  0.625250760  9726.701
## p[8]           0.433999187 0.11703034  0.25000170  0.625250760  9726.701
## p[9]           0.394731327 0.10944983  0.22445594  0.575161345 10247.142
## p[10]          0.394731327 0.10944983  0.22445594  0.575161345 10247.142
## p[11]          0.394731327 0.10944983  0.22445594  0.575161345 10247.142
## p[12]          0.433999187 0.11703034  0.25000170  0.625250760  9726.701
## p[13]          0.458871626 0.12311007  0.27456906  0.668225550  7658.054
## p[14]          0.407815129 0.12023690  0.21942216  0.607378275  7976.795
## p[15]          0.458871626 0.12311007  0.27456906  0.668225550  7658.054
```

```
## p[16]          0.407815129 0.12023690  0.21942216  0.607378275  7976.795
## p[17]          0.407815129 0.12023690  0.21942216  0.607378275  7976.795
## p[18]          0.458871626 0.12311007  0.27456906  0.668225550  7658.054
## p[19]          0.511209815 0.12231598  0.32040864  0.713356280  9760.134
## p[20]          0.405015230 0.11116315  0.23579896  0.589217715 10132.056
## p[21]          0.405015230 0.11116315  0.23579896  0.589217715 10132.056
## p[22]          0.405015230 0.11116315  0.23579896  0.589217715 10132.056
## p[23]          0.511209815 0.12231598  0.32040864  0.713356280  9760.134
## p[24]          0.511209815 0.12231598  0.32040864  0.713356280  9760.134
## p[25]          0.324819411 0.10626442  0.15799183  0.499500925  7885.715
## p[26]          0.324819411 0.10626442  0.15799183  0.499500925  7885.715
## p[27]          0.480357022 0.11758476  0.29478265  0.673799650 10649.245
## p[28]          0.480357022 0.11758476  0.29478265  0.673799650 10649.245
## p[29]          0.324819411 0.10626442  0.15799183  0.499500925  7885.715
## p[30]          0.480357022 0.11758476  0.29478265  0.673799650 10649.245
## p[31]          0.357645023 0.11008530  0.18557661  0.537432410  8428.257
## p[32]          0.608903610 0.13605531  0.38901912  0.826742100  5460.546
## p[33]          0.608903610 0.13605531  0.38901912  0.826742100  5460.546
## p[34]          0.608903610 0.13605531  0.38901912  0.826742100  5460.546
## p[35]          0.357645023 0.11008530  0.18557661  0.537432410  8428.257
## p[36]          0.357645023 0.11008530  0.18557661  0.537432410  8428.257
## p[37]          0.291989964 0.09965676  0.14174076  0.461121385  9680.543
## p[38]          0.291989964 0.09965676  0.14174076  0.461121385  9680.543
## p[39]          0.353097197 0.12482001  0.15791553  0.556616980  5590.441
## p[40]          0.291989964 0.09965676  0.14174076  0.461121385  9680.543
## p[41]          0.291989964 0.09965676  0.14174076  0.461121385  9680.543
## p[42]          0.291989964 0.09965676  0.14174076  0.461121385  9680.543
## p[43]          0.510299912 0.11402546  0.33593351  0.697206095  8969.113
## p[44]          0.365230477 0.11641894  0.18805919  0.563996495  9281.708
## p[45]          0.510299912 0.11402546  0.33593351  0.697206095  8969.113
## p[46]          0.365230477 0.11641894  0.18805919  0.563996495  9281.708
## p[47]          0.510299912 0.11402546  0.33593351  0.697206095  8969.113
## p[48]          0.365230477 0.11641894  0.18805919  0.563996495  9281.708
## p[49]          0.510432258 0.11455822  0.33319322  0.701038735  9074.788
## p[50]          0.344064368 0.11213629  0.17281839  0.532823585 10535.530
## p[51]          0.510432258 0.11455822  0.33319322  0.701038735  9074.788
## p[52]          0.510432258 0.11455822  0.33319322  0.701038735  9074.788
## p[53]          0.510432258 0.11455822  0.33319322  0.701038735  9074.788
## p[54]          0.344064368 0.11213629  0.17281839  0.532823585 10535.530
## p[55]          0.296885452 0.10224713  0.14235618  0.468636390  9136.505
## p[56]          0.463402904 0.12136391  0.27393688  0.662890475  8625.490
## p[57]          0.463402904 0.12136391  0.27393688  0.662890475  8625.490
## p[58]          0.296885452 0.10224713  0.14235618  0.468636390  9136.505
## p[59]          0.296885452 0.10224713  0.14235618  0.468636390  9136.505
## p[60]          0.296885452 0.10224713  0.14235618  0.468636390  9136.505
## p[61]          0.456301450 0.11130606  0.28521530  0.640014605  9383.943
```

```
## p[62]        0.456301450 0.11130606  0.28521530  0.640014605  9383.943
## p[63]        0.456301450 0.11130606  0.28521530  0.640014605  9383.943
## p[64]        0.359217727 0.11759468  0.18267634  0.561288485  9140.799
## p[65]        0.359217727 0.11759468  0.18267634  0.561288485  9140.799
## p[66]        0.456301450 0.11130606  0.28521530  0.640014605  9383.943
## p[67]        0.556543136 0.12327610  0.36822858  0.762661235  7585.245
## p[68]        0.556543136 0.12327610  0.36822858  0.762661235  7585.245
## p[69]        0.325741480 0.11008821  0.15721504  0.511472310  9900.697
## p[70]        0.556543136 0.12327610  0.36822858  0.762661235  7585.245
## p[71]        0.556543136 0.12327610  0.36822858  0.762661235  7585.245
## p[72]        0.325741480 0.11008821  0.15721504  0.511472310  9900.697
## p[73]        0.980099425 0.02281294  0.93857679  0.999076000  6977.265
## p[74]        0.975357113 0.02589081  0.92734400  0.998655990  6423.201
## p[75]        0.975357113 0.02589081  0.92734400  0.998655990  6423.201
## p[76]        0.975357113 0.02589081  0.92734400  0.998655990  6423.201
## p[77]        0.980099425 0.02281294  0.93857679  0.999076000  6977.265
## p[78]        0.980099425 0.02281294  0.93857679  0.999076000  6977.265
## p[79]        0.980030428 0.02323682  0.93804557  0.999071055  6970.058
## p[80]        0.980030428 0.02323682  0.93804557  0.999071055  6970.058
## p[81]        0.980030428 0.02323682  0.93804557  0.999071055  6970.058
## p[82]        0.981917393 0.01938181  0.94570281  0.999012055  6742.529
## p[83]        0.981917393 0.01938181  0.94570281  0.999012055  6742.529
## p[84]        0.981917393 0.01938181  0.94570281  0.999012055  6742.529
## p[85]        0.985843857 0.01561593  0.95723600  0.999327055  6235.107
## p[86]        0.985843857 0.01561593  0.95723600  0.999327055  6235.107
## p[87]        0.977499886 0.02664929  0.92938067  0.998963165  7071.172
## p[88]        0.977499886 0.02664929  0.92938067  0.998963165  7071.172
## p[89]        0.985843857 0.01561593  0.95723600  0.999327055  6235.107
## p[90]        0.977499886 0.02664929  0.92938067  0.998963165  7071.172
## p[91]        0.982653015 0.01852358  0.94746462  0.999105000  6700.299
## p[92]        0.982653015 0.01852358  0.94746462  0.999105000  6700.299
## p[93]        0.985315263 0.01744542  0.95487370  0.999332055  7254.432
## p[94]        0.982653015 0.01852358  0.94746462  0.999105000  6700.299
## p[95]        0.985315263 0.01744542  0.95487370  0.999332055  7254.432
## p[96]        0.985315263 0.01744542  0.95487370  0.999332055  7254.432
## p[97]        0.975015083 0.02720726  0.92528235  0.998641055  6715.467
## p[98]        0.983449466 0.01958548  0.94863497  0.999250275  7562.329
## p[99]        0.975015083 0.02720726  0.92528235  0.998641055  6715.467
## p[100]       0.983449466 0.01958548  0.94863497  0.999250275  7562.329
## p[101]       0.975015083 0.02720726  0.92528235  0.998641055  6715.467
## p[102]       0.983449466 0.01958548  0.94863497  0.999250275  7562.329
## p[103]       0.989901797 0.01274981  0.96691784  0.999607055  6781.465
## p[104]       0.978336554 0.02369639  0.93363407  0.998851165  6542.173
## p[105]       0.989901797 0.01274981  0.96691784  0.999607055  6781.465
## p[106]       0.989901797 0.01274981  0.96691784  0.999607055  6781.465
## p[107]       0.978336554 0.02369639  0.93363407  0.998851165  6542.173
```

```
## p[108]         0.978336554 0.02369639  0.93363407  0.998851165  6542.173
## p[109]         0.976813339 0.02581198  0.92708208  0.998915055  6935.089
## p[110]         0.976813339 0.02581198  0.92708208  0.998915055  6935.089
## p[111]         0.976813339 0.02581198  0.92708208  0.998915055  6935.089
## p[112]         0.979830368 0.02354018  0.93781379  0.999078000  7516.942
## p[113]         0.976813339 0.02581198  0.92708208  0.998915055  6935.089
## p[114]         0.976813339 0.02581198  0.92708208  0.998915055  6935.089
## p[115]         0.976813339 0.02581198  0.92708208  0.998915055  6935.089
## p[116]         0.983143264 0.01962539  0.94679442  0.999246220  7278.490
## p[117]         0.985300847 0.01658805  0.95530195  0.999237000  7155.784
## p[118]         0.983143264 0.01962539  0.94679442  0.999246220  7278.490
## p[119]         0.985300847 0.01658805  0.95530195  0.999237000  7155.784
## p[120]         0.985300847 0.01658805  0.95530195  0.999237000  7155.784
## p[121]         0.985300847 0.01658805  0.95530195  0.999237000  7155.784
## p[122]         0.985367448 0.01623247  0.95474720  0.999224165  6548.123
## p[123]         0.985367448 0.01623247  0.95474720  0.999224165  6548.123
## p[124]         0.981569727 0.02120645  0.94117145  0.999162165  6959.042
## p[125]         0.981569727 0.02120645  0.94117145  0.999162165  6959.042
## p[126]         0.985367448 0.01623247  0.95474720  0.999224165  6548.123
## p[127]         0.985367448 0.01623247  0.95474720  0.999224165  6548.123
## p[128]         0.977413636 0.02554882  0.93157686  0.998916220  6696.989
## p[129]         0.982149398 0.01995603  0.94622264  0.999080055  7044.393
## p[130]         0.977413636 0.02554882  0.93157686  0.998916220  6696.989
## p[131]         0.977413636 0.02554882  0.93157686  0.998916220  6696.989
## p[132]         0.982149398 0.01995603  0.94622264  0.999080055  7044.393
## p[133]         0.977413636 0.02554882  0.93157686  0.998916220  6696.989
## p[134]         0.981977692 0.01961755  0.94615446  0.999039165  6725.990
## p[135]         0.981977692 0.01961755  0.94615446  0.999039165  6725.990
## p[136]         0.981977692 0.01961755  0.94615446  0.999039165  6725.990
## p[137]         0.981977692 0.01961755  0.94615446  0.999039165  6725.990
## p[138]         0.982909761 0.01939168  0.94647875  0.999211000  7281.837
## p[139]         0.981977692 0.01961755  0.94615446  0.999039165  6725.990
## p[140]         0.987545675 0.01446433  0.96168206  0.999421000  6824.919
## p[141]         0.979830368 0.02354018  0.93781379  0.999078000  7516.942
## p[142]         0.987545675 0.01446433  0.96168206  0.999421000  6824.919
## p[143]         0.979830368 0.02354018  0.93781379  0.999078000  7516.942
## p[144]         0.987545675 0.01446433  0.96168206  0.999421000  6824.919
## p[145]         0.269950204 0.09621040  0.12907146  0.432894350  8746.852
## p[146]         0.441020966 0.12701660  0.24674639  0.654677145  8160.219
## p[147]         0.441020966 0.12701660  0.24674639  0.654677145  8160.219
## p[148]         0.441020966 0.12701660  0.24674639  0.654677145  8160.219
## p[149]         0.269950204 0.09621040  0.12907146  0.432894350  8746.852
## p[150]         0.269950204 0.09621040  0.12907146  0.432894350  8746.852
## p[151]         0.441248560 0.12565755  0.24735361  0.650338910  7406.104
## p[152]         0.333081431 0.10426311  0.17760508  0.507244550  8809.163
## p[153]         0.441248560 0.12565755  0.24735361  0.650338910  7406.104
```

```
## p[154]          0.441248560 0.12565755  0.24735361  0.650338910  7406.104
## p[155]          0.333081431 0.10426311  0.17760508  0.507244550  8809.163
## p[156]          0.333081431 0.10426311  0.17760508  0.507244550  8809.163
## p[157]          0.414930024 0.12807566  0.21752279  0.627470995  6539.237
## p[158]          0.394430021 0.12086572  0.21973934  0.603106110  7552.399
## p[159]          0.394430021 0.12086572  0.21973934  0.603106110  7552.399
## p[160]          0.394430021 0.12086572  0.21973934  0.603106110  7552.399
## p[161]          0.414930024 0.12807566  0.21752279  0.627470995  6539.237
## p[162]          0.414930024 0.12807566  0.21752279  0.627470995  6539.237
## p[163]          0.518046188 0.12772891  0.31536479  0.725427495  8761.474
## p[164]          0.342714393 0.10676207  0.18631617  0.526684850  9365.343
## p[165]          0.342714393 0.10676207  0.18631617  0.526684850  9365.343
## p[166]          0.342714393 0.10676207  0.18631617  0.526684850  9365.343
## p[167]          0.518046188 0.12772891  0.31536479  0.725427495  8761.474
## p[168]          0.518046188 0.12772891  0.31536479  0.725427495  8761.474
## p[169]          0.269452101 0.09771565  0.12189384  0.433074480  7425.609
## p[170]          0.487430427 0.12755688  0.28686386  0.696154585  8137.595
## p[171]          0.487430427 0.12755688  0.28686386  0.696154585  8137.595
## p[172]          0.269452101 0.09771565  0.12189384  0.433074480  7425.609
## p[173]          0.269452101 0.09771565  0.12189384  0.433074480  7425.609
## p[174]          0.487430427 0.12755688  0.28686386  0.696154585  8137.595
## p[175]          0.299217453 0.10246340  0.14242409  0.470212355  8106.539
## p[176]          0.299217453 0.10246340  0.14242409  0.470212355  8106.539
## p[177]          0.615143817 0.13786161  0.39055007  0.834144335  5311.754
## p[178]          0.615143817 0.13786161  0.39055007  0.834144335  5311.754
## p[179]          0.299217453 0.10246340  0.14242409  0.470212355  8106.539
## p[180]          0.615143817 0.13786161  0.39055007  0.834144335  5311.754
## p[181]          0.259271610 0.10320371  0.10438494  0.432658135  4735.327
## p[182]          0.212707731 0.09065494  0.08374375  0.369816440  7563.082
## p[183]          0.259271610 0.10320371  0.10438494  0.432658135  4735.327
## p[184]          0.212707731 0.09065494  0.08374375  0.369816440  7563.082
## p[185]          0.259271610 0.10320371  0.10438494  0.432658135  4735.327
## p[186]          0.212707731 0.09065494  0.08374375  0.369816440  7563.082
## p[187]          0.399597774 0.10784948  0.23560002  0.583155560  8109.056
## p[188]          0.399597774 0.10784948  0.23560002  0.583155560  8109.056
## p[189]          0.274283812 0.11233989  0.11369118  0.474774620  7723.958
## p[190]          0.399597774 0.10784948  0.23560002  0.583155560  8109.056
## p[191]          0.399597774 0.10784948  0.23560002  0.583155560  8109.056
## p[192]          0.399597774 0.10784948  0.23560002  0.583155560  8109.056
## p[193]          0.255217647 0.10272470  0.10582854  0.432518865  8148.222
## p[194]          0.400175919 0.11463507  0.22746686  0.596758110  8963.063
## p[195]          0.255217647 0.10272470  0.10582854  0.432518865  8148.222
## p[196]          0.400175919 0.11463507  0.22746686  0.596758110  8963.063
## p[197]          0.400175919 0.11463507  0.22746686  0.596758110  8963.063
## p[198]          0.255217647 0.10272470  0.10582854  0.432518865  8148.222
## p[199]          0.215534250 0.08880571  0.08777558  0.368845520  7915.445
```

```
## p[200]          0.215534250 0.08880571  0.08777558  0.368845520  7915.445
## p[201]          0.215534250 0.08880571  0.08777558  0.368845520  7915.445
## p[202]          0.356707322 0.11584899  0.18212417  0.556069650  8526.870
## p[203]          0.215534250 0.08880571  0.08777558  0.368845520  7915.445
## p[204]          0.215534250 0.08880571  0.08777558  0.368845520  7915.445
## p[205]          0.348896149 0.10323849  0.19271936  0.521560495  9746.901
## p[206]          0.348896149 0.10323849  0.19271936  0.521560495  9746.901
## p[207]          0.269258669 0.11305717  0.10923685  0.467571620  7302.503
## p[208]          0.348896149 0.10323849  0.19271936  0.521560495  9746.901
## p[209]          0.348896149 0.10323849  0.19271936  0.521560495  9746.901
## p[210]          0.348896149 0.10323849  0.19271936  0.521560495  9746.901
## p[211]          0.239108108 0.09660858  0.09932121  0.406835660  8509.430
## p[212]          0.239108108 0.09660858  0.09932121  0.406835660  8509.430
## p[213]          0.239108108 0.09660858  0.09932121  0.406835660  8509.430
## p[214]          0.446653718 0.13306919  0.25176607  0.676758735  7712.378
## p[215]          0.239108108 0.09660858  0.09932121  0.406835660  8509.430
## p[216]          0.239108108 0.09660858  0.09932121  0.406835660  8509.430
## p[217]          0.287926656 0.09840402  0.13971106  0.450310155  9051.795
## p[218]          0.399047747 0.11883580  0.21888251  0.601375675  9655.475
## p[219]          0.287926656 0.09840402  0.13971106  0.450310155  9051.795
## p[220]          0.287926656 0.09840402  0.13971106  0.450310155  9051.795
## p[221]          0.399047747 0.11883580  0.21888251  0.601375675  9655.475
## p[222]          0.399047747 0.11883580  0.21888251  0.601375675  9655.475
## p[223]          0.352900628 0.10464587  0.19901921  0.530783210  8563.730
## p[224]          0.399281532 0.11864911  0.21469712  0.597818430  8599.787
## p[225]          0.352900628 0.10464587  0.19901921  0.530783210  8563.730
## p[226]          0.399281532 0.11864911  0.21469712  0.597818430  8599.787
## p[227]          0.352900628 0.10464587  0.19901921  0.530783210  8563.730
## p[228]          0.399281532 0.11864911  0.21469712  0.597818430  8599.787
## p[229]          0.373854060 0.11908576  0.19026235  0.572220245  7457.040
## p[230]          0.415672375 0.12061427  0.23738219  0.623892365  7278.042
## p[231]          0.373854060 0.11908576  0.19026235  0.572220245  7457.040
## p[232]          0.373854060 0.11908576  0.19026235  0.572220245  7457.040
## p[233]          0.415672375 0.12061427  0.23738219  0.623892365  7278.042
## p[234]          0.415672375 0.12061427  0.23738219  0.623892365  7278.042
## p[235]          0.363069045 0.10818339  0.20361025  0.547080885  9541.599
## p[236]          0.475243959 0.12379775  0.28393199  0.681540875  9807.349
## p[237]          0.363069045 0.10818339  0.20361025  0.547080885  9541.599
## p[238]          0.475243959 0.12379775  0.28393199  0.681540875  9807.349
## p[239]          0.363069045 0.10818339  0.20361025  0.547080885  9541.599
## p[240]          0.475243959 0.12379775  0.28393199  0.681540875  9807.349
## p[241]          0.444544997 0.12047013  0.25759129  0.644734860  9457.591
## p[242]          0.444544997 0.12047013  0.25759129  0.644734860  9457.591
## p[243]          0.286999614 0.09951332  0.13570467  0.454718270  8184.113
## p[244]          0.286999614 0.09951332  0.13570467  0.454718270  8184.113
## p[245]          0.444544997 0.12047013  0.25759129  0.644734860  9457.591
```

```
## p[246]         0.286999614 0.09951332  0.13570467  0.454718270  8184.113
## p[247]         0.318174894 0.10524811  0.15789878  0.495169320  8824.674
## p[248]         0.318174894 0.10524811  0.15789878  0.495169320  8824.674
## p[249]         0.575005331 0.14038033  0.34968067  0.803018300  5871.399
## p[250]         0.575005331 0.14038033  0.34968067  0.803018300  5871.399
## p[251]         0.318174894 0.10524811  0.15789878  0.495169320  8824.674
## p[252]         0.575005331 0.14038033  0.34968067  0.803018300  5871.399
## p[253]         0.287232005 0.11076541  0.12318695  0.476708745  5326.859
## p[254]         0.194095606 0.08829048  0.06672906  0.346400225  6652.222
## p[255]         0.287232005 0.11076541  0.12318695  0.476708745  5326.859
## p[256]         0.194095606 0.08829048  0.06672906  0.346400225  6652.222
## p[257]         0.194095606 0.08829048  0.06672906  0.346400225  6652.222
## p[258]         0.194095606 0.08829048  0.06672906  0.346400225  6652.222
## p[259]         0.434270584 0.11160733  0.26911888  0.622641180  7883.189
## p[260]         0.434270584 0.11160733  0.26911888  0.622641180  7883.189
## p[261]         0.252526429 0.11351545  0.09164363  0.452614985  6623.263
## p[262]         0.434270584 0.11160733  0.26911888  0.622641180  7883.189
## p[263]         0.434270584 0.11160733  0.26911888  0.622641180  7883.189
## p[264]         0.434270584 0.11160733  0.26911888  0.622641180  7883.189
## p[265]         0.234191601 0.10175038  0.08512392  0.411166805  7323.746
## p[266]         0.234191601 0.10175038  0.08512392  0.411166805  7323.746
## p[267]         0.434661041 0.11726565  0.26000163  0.633478250  8561.499
## p[268]         0.234191601 0.10175038  0.08512392  0.411166805  7323.746
## p[269]         0.434661041 0.11726565  0.26000163  0.633478250  8561.499
## p[270]         0.434661041 0.11726565  0.26000163  0.633478250  8561.499
## p[271]         0.197130640 0.08787061  0.07029265  0.348663365  6921.304
## p[272]         0.197130640 0.08787061  0.07029265  0.348663365  6921.304
## p[273]         0.197130640 0.08787061  0.07029265  0.348663365  6921.304
## p[274]         0.197130640 0.08787061  0.07029265  0.348663365  6921.304
## p[275]         0.389505354 0.11916410  0.21048099  0.593025505  8408.903
## p[276]         0.197130640 0.08787061  0.07029265  0.348663365  6921.304
## p[277]         0.381969642 0.10679999  0.21890116  0.561739635  9386.846
## p[278]         0.247569397 0.11213061  0.09057867  0.445732380  5979.944
## p[279]         0.381969642 0.10679999  0.21890116  0.561739635  9386.846
## p[280]         0.381969642 0.10679999  0.21890116  0.561739635  9386.846
## p[281]         0.381969642 0.10679999  0.21890116  0.561739635  9386.846
## p[282]         0.381969642 0.10679999  0.21890116  0.561739635  9386.846
## p[283]         0.481298805 0.13218243  0.28382775  0.706432185  7709.431
## p[284]         0.218872308 0.09501705  0.08121706  0.382930290  7292.278
## p[285]         0.218872308 0.09501705  0.08121706  0.382930290  7292.278
## p[286]         0.481298805 0.13218243  0.28382775  0.706432185  7709.431
## p[287]         0.218872308 0.09501705  0.08121706  0.382930290  7292.278
## p[288]         0.218872308 0.09501705  0.08121706  0.382930290  7292.278
## p[289]         0.455310392 0.11978027  0.26997416  0.652116055 10562.576
## p[290]         0.325436083 0.10252175  0.16745330  0.492900010  9655.648
## p[291]         0.325436083 0.10252175  0.16745330  0.492900010  9655.648
```

```
## p[292]          0.325436083 0.10252175  0.16745330  0.492900010  9655.648
## p[293]          0.455310392 0.11978027  0.26997416  0.652116055 10562.576
## p[294]          0.455310392 0.11978027  0.26997416  0.652116055 10562.576
## p[295]          0.394778484 0.10868557  0.22624874  0.576278390  9541.006
## p[296]          0.394778484 0.10868557  0.22624874  0.576278390  9541.006
## p[297]          0.455459025 0.11972980  0.26590143  0.649321165  8785.988
## p[298]          0.455459025 0.11972980  0.26590143  0.649321165  8785.988
## p[299]          0.394778484 0.10868557  0.22624874  0.576278390  9541.006
## p[300]          0.455459025 0.11972980  0.26590143  0.649321165  8785.988
## p[301]          0.428955049 0.12381151  0.23253635  0.628565660  7529.851
## p[302]          0.428955049 0.12381151  0.23253635  0.628565660  7529.851
## p[303]          0.459004973 0.12296861  0.27388552  0.666614825  8283.190
## p[304]          0.428955049 0.12381151  0.23253635  0.628565660  7529.851
## p[305]          0.459004973 0.12296861  0.27388552  0.666614825  8283.190
## p[306]          0.459004973 0.12296861  0.27388552  0.666614825  8283.190
## p[307]          0.532833724 0.12159678  0.33915730  0.730678620 10075.400
## p[308]          0.405226751 0.11311339  0.23445573  0.595421055 11469.272
## p[309]          0.532833724 0.12159678  0.33915730  0.730678620 10075.400
## p[310]          0.532833724 0.12159678  0.33915730  0.730678620 10075.400
## p[311]          0.405226751 0.11311339  0.23445573  0.595421055 11469.272
## p[312]          0.405226751 0.11311339  0.23445573  0.595421055 11469.272
## p[313]          0.501977121 0.11952630  0.31130903  0.696024280  9197.106
## p[314]          0.501977121 0.11952630  0.31130903  0.696024280  9197.106
## p[315]          0.501977121 0.11952630  0.31130903  0.696024280  9197.106
## p[316]          0.324903552 0.10594293  0.15940592  0.498581880  8383.393
## p[317]          0.324903552 0.10594293  0.15940592  0.498581880  8383.393
## p[318]          0.324903552 0.10594293  0.15940592  0.498581880  8383.393
## p[319]          0.629193228 0.13307403  0.41063768  0.837664430  5544.652
## p[320]          0.629193228 0.13307403  0.41063768  0.837664430  5544.652
## p[321]          0.357673485 0.10950088  0.18774061  0.539539525  9634.226
## p[322]          0.629193228 0.13307403  0.41063768  0.837664430  5544.652
## p[323]          0.357673485 0.10950088  0.18774061  0.539539525  9634.226
## p[324]          0.357673485 0.10950088  0.18774061  0.539539525  9634.226
## p[325]          0.340088419 0.11738326  0.15433945  0.530919510  5780.013
## p[326]          0.287213112 0.10169527  0.13415443  0.459839320  8211.762
## p[327]          0.340088419 0.11738326  0.15433945  0.530919510  5780.013
## p[328]          0.287213112 0.10169527  0.13415443  0.459839320  8211.762
## p[329]          0.287213112 0.10169527  0.13415443  0.459839320  8211.762
## p[330]          0.340088419 0.11738326  0.15433945  0.530919510  5780.013
## p[331]          0.359363863 0.11749942  0.18294421  0.557016750  8604.028
## p[332]          0.496973879 0.11318004  0.32012233  0.681669510  7538.694
## p[333]          0.359363863 0.11749942  0.18294421  0.557016750  8604.028
## p[334]          0.496973879 0.11318004  0.32012233  0.681669510  7538.694
## p[335]          0.359363863 0.11749942  0.18294421  0.557016750  8604.028
## p[336]          0.496973879 0.11318004  0.32012233  0.681669510  7538.694
## p[337]          0.337906731 0.11115070  0.16966934  0.524891935  9442.540
```

```
## p[338]         0.497051877 0.11654029  0.31616275  0.687731540  8478.922
## p[339]         0.497051877 0.11654029  0.31616275  0.687731540  8478.922
## p[340]         0.337906731 0.11115070  0.16966934  0.524891935  9442.540
## p[341]         0.337906731 0.11115070  0.16966934  0.524891935  9442.540
## p[342]         0.497051877 0.11654029  0.31616275  0.687731540  8478.922
## p[343]         0.449995011 0.11760302  0.26463422  0.643990975  8555.226
## p[344]         0.449995011 0.11760302  0.26463422  0.643990975  8555.226
## p[345]         0.291523097 0.10210177  0.13673194  0.463904705  9069.725
## p[346]         0.291523097 0.10210177  0.13673194  0.463904705  9069.725
## p[347]         0.291523097 0.10210177  0.13673194  0.463904705  9069.725
## p[348]         0.449995011 0.11760302  0.26463422  0.643990975  8555.226
## p[349]         0.353055875 0.11511320  0.17606384  0.551233415  8701.688
## p[350]         0.443034969 0.11177478  0.26808145  0.626706570  8981.514
## p[351]         0.443034969 0.11177478  0.26808145  0.626706570  8981.514
## p[352]         0.353055875 0.11511320  0.17606384  0.551233415  8701.688
## p[353]         0.353055875 0.11511320  0.17606384  0.551233415  8701.688
## p[354]         0.353055875 0.11511320  0.17606384  0.551233415  8701.688
## p[355]         0.543334441 0.12564870  0.34632589  0.751128200  6776.954
## p[356]         0.543334441 0.12564870  0.34632589  0.751128200  6776.954
## p[357]         0.319968735 0.10959483  0.15247189  0.502593275  9500.362
## p[358]         0.319968735 0.10959483  0.15247189  0.502593275  9500.362
## p[359]         0.543334441 0.12564870  0.34632589  0.751128200  6776.954
## p[360]         0.543334441 0.12564870  0.34632589  0.751128200  6776.954
## p[361]         0.609568323 0.11899954  0.40720158  0.788407560  9565.954
## p[362]         0.623985771 0.11433995  0.43264746  0.801130685  8241.537
## p[363]         0.623985771 0.11433995  0.43264746  0.801130685  8241.537
## p[364]         0.609568323 0.11899954  0.40720158  0.788407560  9565.954
## p[365]         0.623985771 0.11433995  0.43264746  0.801130685  8241.537
## p[366]         0.609568323 0.11899954  0.40720158  0.788407560  9565.954
## p[367]         0.609893002 0.11737604  0.40938870  0.786493815  9266.337
## p[368]         0.609893002 0.11737604  0.40938870  0.786493815  9266.337
## p[369]         0.609893002 0.11737604  0.40938870  0.786493815  9266.337
## p[370]         0.692316259 0.10113855  0.52310072  0.848223135  8484.723
## p[371]         0.692316259 0.10113855  0.52310072  0.848223135  8484.723
## p[372]         0.692316259 0.10113855  0.52310072  0.848223135  8484.723
## p[373]         0.743335850 0.09719602  0.58219192  0.891983705  6756.937
## p[374]         0.743335850 0.09719602  0.58219192  0.891983705  6756.937
## p[375]         0.743335850 0.09719602  0.58219192  0.891983705  6756.937
## p[376]         0.583287873 0.12400257  0.36915539  0.766879590  9090.716
## p[377]         0.583287873 0.12400257  0.36915539  0.766879590  9090.716
## p[378]         0.583287873 0.12400257  0.36915539  0.766879590  9090.716
## p[379]         0.701024178 0.10083017  0.53126287  0.854858510  8454.374
## p[380]         0.680116218 0.11068899  0.49039091  0.844364155  8503.644
## p[381]         0.701024178 0.10083017  0.53126287  0.854858510  8454.374
## p[382]         0.680116218 0.11068899  0.49039091  0.844364155  8503.644
## p[383]         0.701024178 0.10083017  0.53126287  0.854858510  8454.374
```

```
## p[384]          0.680116218 0.11068899  0.49039091  0.844364155  8503.644
## p[385]          0.653138489 0.11268243  0.46310826  0.821663715  9944.700
## p[386]          0.622217160 0.11785376  0.41787992  0.798288290  7928.024
## p[387]          0.622217160 0.11785376  0.41787992  0.798288290  7928.024
## p[388]          0.653138489 0.11268243  0.46310826  0.821663715  9944.700
## p[389]          0.622217160 0.11785376  0.41787992  0.798288290  7928.024
## p[390]          0.653138489 0.11268243  0.46310826  0.821663715  9944.700
## p[391]          0.656371279 0.11267019  0.46548854  0.822621475  8023.551
## p[392]          0.656371279 0.11267019  0.46548854  0.822621475  8023.551
## p[393]          0.757538981 0.10946757  0.56913391  0.913109235  5121.909
## p[394]          0.757538981 0.10946757  0.56913391  0.913109235  5121.909
## p[395]          0.757538981 0.10946757  0.56913391  0.913109235  5121.909
## p[396]          0.656371279 0.11267019  0.46548854  0.822621475  8023.551
## p[397]          0.524421268 0.13026264  0.29923321  0.716401640  6466.751
## p[398]          0.541127736 0.12020141  0.33981073  0.725848905  9432.487
## p[399]          0.524421268 0.13026264  0.29923321  0.716401640  6466.751
## p[400]          0.524421268 0.13026264  0.29923321  0.716401640  6466.751
## p[401]          0.541127736 0.12020141  0.33981073  0.725848905  9432.487
## p[402]          0.541127736 0.12020141  0.33981073  0.725848905  9432.487
## p[403]          0.620285891 0.11718935  0.42525792  0.800093030  9695.100
## p[404]          0.620285891 0.11718935  0.42525792  0.800093030  9695.100
## p[405]          0.681507986 0.10128195  0.50911555  0.834600775  7849.155
## p[406]          0.681507986 0.10128195  0.50911555  0.834600775  7849.155
## p[407]          0.681507986 0.10128195  0.50911555  0.834600775  7849.155
## p[408]          0.681507986 0.10128195  0.50911555  0.834600775  7849.155
## p[409]          0.681000514 0.10452330  0.50248293  0.837806985  8099.896
## p[410]          0.681000514 0.10452330  0.50248293  0.837806985  8099.896
## p[411]          0.598814832 0.11641628  0.40135139  0.774847260 10564.387
## p[412]          0.681000514 0.10452330  0.50248293  0.837806985  8099.896
## p[413]          0.598814832 0.11641628  0.40135139  0.774847260 10564.387
## p[414]          0.598814832 0.11641628  0.40135139  0.774847260 10564.387
## p[415]          0.546475338 0.12039761  0.34146227  0.728896290  8596.091
## p[416]          0.546475338 0.12039761  0.34146227  0.728896290  8596.091
## p[417]          0.638092808 0.11391395  0.44307313  0.808862925  9568.252
## p[418]          0.638092808 0.11391395  0.44307313  0.808862925  9568.252
## p[419]          0.546475338 0.12039761  0.34146227  0.728896290  8596.091
## p[420]          0.638092808 0.11391395  0.44307313  0.808862925  9568.252
## p[421]          0.632826352 0.10814578  0.44735040  0.795135615  9042.411
## p[422]          0.632826352 0.10814578  0.44735040  0.795135615  9042.411
## p[423]          0.614149913 0.11573990  0.42390975  0.793083220  9036.279
## p[424]          0.614149913 0.11573990  0.42390975  0.793083220  9036.279
## p[425]          0.614149913 0.11573990  0.42390975  0.793083220  9036.279
## p[426]          0.632826352 0.10814578  0.44735040  0.795135615  9042.411
## p[427]          0.578822195 0.11884819  0.37236717  0.755324935  9135.127
## p[428]          0.578822195 0.11884819  0.37236717  0.755324935  9135.127
## p[429]          0.717441316 0.10940376  0.53110001  0.880794695  6661.052
```

```
## p[430]        0.578822195 0.11884819  0.37236717  0.755324935  9135.127
## p[431]        0.717441316 0.10940376  0.53110001  0.880794695  6661.052
## p[432]        0.578822195 0.11884819  0.37236717  0.755324935  9135.127
## p[433]        0.856902043 0.07523528  0.71722351  0.950956440  8088.728
## p[434]        0.818605624 0.08718859  0.65924094  0.928504550  8763.819
## p[435]        0.818605624 0.08718859  0.65924094  0.928504550  8763.819
## p[436]        0.856902043 0.07523528  0.71722351  0.950956440  8088.728
## p[437]        0.856902043 0.07523528  0.71722351  0.950956440  8088.728
## p[438]        0.818605624 0.08718859  0.65924094  0.928504550  8763.819
## p[439]        0.859842093 0.07046813  0.73166889  0.947816585  9529.343
## p[440]        0.859842093 0.07046813  0.73166889  0.947816585  9529.343
## p[441]        0.857140105 0.07578208  0.71360548  0.949010495  9328.773
## p[442]        0.859842093 0.07046813  0.73166889  0.947816585  9529.343
## p[443]        0.857140105 0.07578208  0.71360548  0.949010495  9328.773
## p[444]        0.857140105 0.07578208  0.71360548  0.949010495  9328.773
## p[445]        0.842594333 0.08298708  0.68881903  0.943928495  8920.410
## p[446]        0.842594333 0.08298708  0.68881903  0.943928495  8920.410
## p[447]        0.887636026 0.06142739  0.77566962  0.962785650  7392.919
## p[448]        0.887636026 0.06142739  0.77566962  0.962785650  7392.919
## p[449]        0.887636026 0.06142739  0.77566962  0.962785650  7392.919
## p[450]        0.842594333 0.08298708  0.68881903  0.943928495  8920.410
## p[451]        0.864955753 0.06796827  0.74331567  0.949905730  8855.306
## p[452]        0.890427797 0.06241240  0.77381396  0.966001155  7738.950
## p[453]        0.864955753 0.06796827  0.74331567  0.949905730  8855.306
## p[454]        0.864955753 0.06796827  0.74331567  0.949905730  8855.306
## p[455]        0.890427797 0.06241240  0.77381396  0.966001155  7738.950
## p[456]        0.890427797 0.06241240  0.77381396  0.966001155  7738.950
## p[457]        0.878578405 0.06617597  0.75514718  0.959855385  8123.499
## p[458]        0.817138295 0.08963116  0.65239695  0.928067495  8014.875
## p[459]        0.817138295 0.08963116  0.65239695  0.928067495  8014.875
## p[460]        0.817138295 0.08963116  0.65239695  0.928067495  8014.875
## p[461]        0.878578405 0.06617597  0.75514718  0.959855385  8123.499
## p[462]        0.878578405 0.06617597  0.75514718  0.959855385  8123.499
## p[463]        0.921732527 0.05246502  0.82387694  0.981697165  5415.568
## p[464]        0.838143619 0.08181926  0.68938839  0.939141730  8605.396
## p[465]        0.838143619 0.08181926  0.68938839  0.939141730  8605.396
## p[466]        0.838143619 0.08181926  0.68938839  0.939141730  8605.396
## p[467]        0.921732527 0.05246502  0.82387694  0.981697165  5415.568
## p[468]        0.921732527 0.05246502  0.82387694  0.981697165  5415.568
## p[469]        0.855456173 0.08215356  0.70497146  0.956817760  6333.051
## p[470]        0.875476919 0.07460342  0.73846772  0.966965660  5708.521
## p[471]        0.855456173 0.08215356  0.70497146  0.956817760  6333.051
## p[472]        0.875476919 0.07460342  0.73846772  0.966965660  5708.521
## p[473]        0.855456173 0.08215356  0.70497146  0.956817760  6333.051
## p[474]        0.855456173 0.08215356  0.70497146  0.956817760  6333.051
## p[475]        0.921463209 0.04580984  0.83828817  0.978688440  6677.200
```

```
## p[476]        0.907910841 0.05658530  0.80571853  0.976233440  6830.750
## p[477]        0.921463209 0.04580984  0.83828817  0.978688440  6677.200
## p[478]        0.907910841 0.05658530  0.80571853  0.976233440  6830.750
## p[479]        0.921463209 0.04580984  0.83828817  0.978688440  6677.200
## p[480]        0.907910841 0.05658530  0.80571853  0.976233440  6830.750
## p[481]        0.899438762 0.06145512  0.78746923  0.974290860  6728.570
## p[482]        0.921247684 0.04604822  0.83762834  0.979315705  7079.509
## p[483]        0.921247684 0.04604822  0.83762834  0.979315705  7079.509
## p[484]        0.899438762 0.06145512  0.78746923  0.974290860  6728.570
## p[485]        0.899438762 0.06145512  0.78746923  0.974290860  6728.570
## p[486]        0.921247684 0.04604822  0.83762834  0.979315705  7079.509
## p[487]        0.878391836 0.07156885  0.74582621  0.967637650  5781.143
## p[488]        0.878391836 0.07156885  0.74582621  0.967637650  5781.143
## p[489]        0.905361515 0.05599081  0.80539631  0.975030870  7326.452
## p[490]        0.878391836 0.07156885  0.74582621  0.967637650  5781.143
## p[491]        0.878391836 0.07156885  0.74582621  0.967637650  5781.143
## p[492]        0.878391836 0.07156885  0.74582621  0.967637650  5781.143
## p[493]        0.904317194 0.05350966  0.80803679  0.973201210  7362.002
## p[494]        0.905326994 0.05809005  0.79757736  0.976411330  6498.532
## p[495]        0.905326994 0.05809005  0.79757736  0.976411330  6498.532
## p[496]        0.904317194 0.05350966  0.80803679  0.973201210  7362.002
## p[497]        0.904317194 0.05350966  0.80803679  0.973201210  7362.002
## p[498]        0.904317194 0.05350966  0.80803679  0.973201210  7362.002
## p[499]        0.932323241 0.04375364  0.85125313  0.984573055  6405.586
## p[500]        0.932323241 0.04375364  0.85125313  0.984573055  6405.586
## p[501]        0.932323241 0.04375364  0.85125313  0.984573055  6405.586
## p[502]        0.891082902 0.06794858  0.76854585  0.972196060  5902.790
## p[503]        0.891082902 0.06794858  0.76854585  0.972196060  5902.790
## p[504]        0.891082902 0.06794858  0.76854585  0.972196060  5902.790
## Rho_A[1,1]    1.000000000 0.00000000  1.00000000  1.000000000        NaN
## Rho_A[1,2]    0.008494181 0.29831040 -0.47219382  0.491518995  8747.145
## Rho_A[1,3]    0.043365503 0.30182758 -0.45205797  0.518570335  7514.347
## Rho_A[1,4]    0.004980725 0.30234721 -0.48341318  0.481705465 10722.785
## Rho_A[2,1]    0.008494181 0.29831040 -0.47219382  0.491518995  8747.145
## Rho_A[2,2]    1.000000000 0.00000000  1.00000000  1.000000000        NaN
## Rho_A[2,3]   -0.054183695 0.29991824 -0.53145255  0.437958380  7201.983
## Rho_A[2,4]    0.033232624 0.30405544 -0.46089486  0.521944330  8524.996
## Rho_A[3,1]    0.043365503 0.30182758 -0.45205797  0.518570335  7514.347
## Rho_A[3,2]   -0.054183695 0.29991824 -0.53145255  0.437958380  7201.983
## Rho_A[3,3]    1.000000000 0.00000000  1.00000000  1.000000000        NaN
## Rho_A[3,4]    0.040226990 0.29989094 -0.44883877  0.514412080  7538.227
## Rho_A[4,1]    0.004980725 0.30234721 -0.48341318  0.481705465 10722.785
## Rho_A[4,2]    0.033232624 0.30405544 -0.46089486  0.521944330  8524.996
## Rho_A[4,3]    0.040226990 0.29989094 -0.44883877  0.514412080  7538.227
## Rho_A[4,4]    1.000000000 0.00000000  1.00000000  1.000000000        NaN
## Rho_B[1,1]    1.000000000 0.00000000  1.00000000  1.000000000        NaN
```

```
## Rho_B[1,2]   -0.053082664 0.29859316 -0.52635807  0.440238935  6974.250
## Rho_B[1,3]    0.007164639 0.29808511 -0.46744963  0.478663615 10006.219
## Rho_B[1,4]    0.027543456 0.29385578 -0.44019707  0.495376280  8792.906
## Rho_B[2,1]   -0.053082664 0.29859316 -0.52635807  0.440238935  6974.250
## Rho_B[2,2]    1.000000000 0.00000000  1.00000000  1.000000000       NaN
## Rho_B[2,3]   -0.049089531 0.29831772 -0.52142488  0.440885300  8736.300
## Rho_B[2,4]    0.059589718 0.29908642 -0.42446376  0.532439400  7905.825
## Rho_B[3,1]    0.007164639 0.29808511 -0.46744963  0.478663615 10006.219
## Rho_B[3,2]   -0.049089531 0.29831772 -0.52142488  0.440885300  8736.300
## Rho_B[3,3]    1.000000000 0.00000000  1.00000000  1.000000000       NaN
## Rho_B[3,4]   -0.006867794 0.29435179 -0.47955287  0.465430575  7942.174
## Rho_B[4,1]    0.027543456 0.29385578 -0.44019707  0.495376280  8792.906
## Rho_B[4,2]    0.059589718 0.29908642 -0.42446376  0.532439400  7905.825
## Rho_B[4,3]   -0.006867794 0.29435179 -0.47955287  0.465430575  7942.174
## Rho_B[4,4]    1.000000000 0.00000000  1.00000000  1.000000000       NaN
##                  Rhat4
## zA[1,1]      1.0003735
## zA[1,2]      0.9997625
## zA[1,3]      1.0000932
## zA[1,4]      1.0001855
## zA[1,5]      0.9998868
## zA[1,6]      1.0003146
## zA[1,7]      1.0002259
## zA[2,1]      0.9997864
## zA[2,2]      0.9997160
## zA[2,3]      1.0003444
## zA[2,4]      1.0000507
## zA[2,5]      0.9998780
## zA[2,6]      1.0000047
## zA[2,7]      0.9998091
## zA[3,1]      0.9998199
## zA[3,2]      0.9997381
## zA[3,3]      0.9999074
## zA[3,4]      0.9996316
## zA[3,5]      1.0002866
## zA[3,6]      1.0001846
## zA[3,7]      0.9997046
## zA[4,1]      0.9997478
## zA[4,2]      0.9999807
## zA[4,3]      0.9999843
## zA[4,4]      0.9999101
## zA[4,5]      0.9999729
## zA[4,6]      0.9996320
## zA[4,7]      0.9997893
## zB[1,1]      0.9997689
## zB[1,2]      1.0000275
```

```
## zB[1,3]       1.0002360
## zB[1,4]       1.0000497
## zB[1,5]       0.9998485
## zB[1,6]       1.0001872
## zB[2,1]       0.9999302
## zB[2,2]       1.0001288
## zB[2,3]       1.0000893
## zB[2,4]       0.9997703
## zB[2,5]       0.9998038
## zB[2,6]       0.9998899
## zB[3,1]       1.0000076
## zB[3,2]       0.9996228
## zB[3,3]       1.0002856
## zB[3,4]       1.0000895
## zB[3,5]       0.9999848
## zB[3,6]       1.0000501
## zB[4,1]       0.9999347
## zB[4,2]       1.0001342
## zB[4,3]       0.9997336
## zB[4,4]       0.9998985
## zB[4,5]       1.0003051
## zB[4,6]       1.0001899
## zAbar[1]      1.0000756
## zAbar[2]      1.0000788
## zAbar[3]      0.9998530
## zAbar[4]      0.9996531
## zAbar[5]      0.9998688
## zAbar[6]      1.0007422
## zAbar[7]      1.0003178
## zBbar[1]      0.9997531
## zBbar[2]      0.9997422
## zBbar[3]      0.9999256
## zBbar[4]      0.9999270
## zBbar[5]      0.9997675
## zBbar[6]      0.9996527
## tau_A         1.0007667
## tau_B         1.0012401
## sigma_A[1]    1.0000445
## sigma_A[2]    1.0003676
## sigma_A[3]    1.0000211
## sigma_A[4]    1.0000531
## sigma_B[1]    1.0003873
## sigma_B[2]    1.0006678
## sigma_B[3]    0.9997056
## sigma_B[4]    1.0004480
## L_Rho_A[1,1]        NaN
```

```
## L_Rho_A[1,2]       NaN
## L_Rho_A[1,3]       NaN
## L_Rho_A[1,4]       NaN
## L_Rho_A[2,1] 0.9998256
## L_Rho_A[2,2] 1.0002309
## L_Rho_A[2,3]       NaN
## L_Rho_A[2,4]       NaN
## L_Rho_A[3,1] 0.9999680
## L_Rho_A[3,2] 0.9997912
## L_Rho_A[3,3] 1.0000743
## L_Rho_A[3,4]       NaN
## L_Rho_A[4,1] 0.9999356
## L_Rho_A[4,2] 0.9996793
## L_Rho_A[4,3] 0.9996947
## L_Rho_A[4,4] 1.0005552
## L_Rho_B[1,1]       NaN
## L_Rho_B[1,2]       NaN
## L_Rho_B[1,3]       NaN
## L_Rho_B[1,4]       NaN
## L_Rho_B[2,1] 1.0004023
## L_Rho_B[2,2] 1.0005927
## L_Rho_B[2,3]       NaN
## L_Rho_B[2,4]       NaN
## L_Rho_B[3,1] 0.9999085
## L_Rho_B[3,2] 0.9996951
## L_Rho_B[3,3] 1.0003560
## L_Rho_B[3,4]       NaN
## L_Rho_B[4,1] 0.9998687
## L_Rho_B[4,2] 1.0000610
## L_Rho_B[4,3] 1.0004146
## L_Rho_B[4,4] 1.0016175
## a[1,1]       1.0003133
## a[1,2]       1.0003690
## a[1,3]       1.0001947
## a[1,4]       0.9997848
## a[2,1]       0.9998989
## a[2,2]       1.0000524
## a[2,3]       0.9997270
## a[2,4]       1.0009472
## a[3,1]       1.0006784
## a[3,2]       1.0006500
## a[3,3]       1.0005409
## a[3,4]       0.9999228
## a[4,1]       1.0005310
## a[4,2]       1.0001605
## a[4,3]       1.0002799
```

```
## a[4,4]        1.0001587
## a[5,1]        1.0000118
## a[5,2]        1.0002550
## a[5,3]        1.0005072
## a[5,4]        0.9998373
## a[6,1]        1.0004207
## a[6,2]        0.9998571
## a[6,3]        0.9999897
## a[6,4]        1.0002323
## a[7,1]        1.0002897
## a[7,2]        0.9996670
## a[7,3]        0.9997811
## a[7,4]        1.0003260
## b[1,1]        0.9998153
## b[1,2]        1.0001972
## b[1,3]        1.0000839
## b[1,4]        1.0001130
## b[2,1]        0.9997745
## b[2,2]        0.9999693
## b[2,3]        1.0000369
## b[2,4]        1.0000404
## b[3,1]        1.0003654
## b[3,2]        1.0002350
## b[3,3]        0.9997420
## b[3,4]        1.0008090
## b[4,1]        1.0000051
## b[4,2]        0.9996738
## b[4,3]        1.0000815
## b[4,4]        0.9998490
## b[5,1]        0.9998465
## b[5,2]        1.0004455
## b[5,3]        1.0001435
## b[5,4]        0.9999695
## b[6,1]        1.0001671
## b[6,2]        1.0002070
## b[6,3]        1.0000618
## b[6,4]        1.0005968
## abar[1]       1.0004418
## abar[2]       1.0000966
## abar[3]       1.0003798
## abar[4]       1.0002549
## abar[5]       1.0001411
## abar[6]       1.0005492
## abar[7]       0.9997363
## bbar[1]       0.9999637
## bbar[2]       1.0001531
```

```
## bbar[3]       1.0005598
## bbar[4]       1.0001790
## bbar[5]       1.0000228
## bbar[6]       1.0000745
## p[1]          0.9999506
## p[2]          0.9999506
## p[3]          0.9997383
## p[4]          0.9999506
## p[5]          0.9997383
## p[6]          0.9997383
## p[7]          0.9999348
## p[8]          0.9999348
## p[9]          0.9997159
## p[10]         0.9997159
## p[11]         0.9997159
## p[12]         0.9999348
## p[13]         0.9999109
## p[14]         1.0003205
## p[15]         0.9999109
## p[16]         1.0003205
## p[17]         1.0003205
## p[18]         0.9999109
## p[19]         0.9995995
## p[20]         0.9998229
## p[21]         0.9998229
## p[22]         0.9998229
## p[23]         0.9995995
## p[24]         0.9995995
## p[25]         1.0000472
## p[26]         1.0000472
## p[27]         0.9998199
## p[28]         0.9998199
## p[29]         1.0000472
## p[30]         0.9998199
## p[31]         1.0002269
## p[32]         0.9998666
## p[33]         0.9998666
## p[34]         0.9998666
## p[35]         1.0002269
## p[36]         1.0002269
## p[37]         0.9997944
## p[38]         0.9997944
## p[39]         1.0001953
## p[40]         0.9997944
## p[41]         0.9997944
## p[42]         0.9997944
```

```
## p[43]          0.9999487
## p[44]          0.9998172
## p[45]          0.9999487
## p[46]          0.9998172
## p[47]          0.9999487
## p[48]          0.9998172
## p[49]          1.0000114
## p[50]          0.9997720
## p[51]          1.0000114
## p[52]          1.0000114
## p[53]          1.0000114
## p[54]          0.9997720
## p[55]          0.9998401
## p[56]          0.9998692
## p[57]          0.9998692
## p[58]          0.9998401
## p[59]          0.9998401
## p[60]          0.9998401
## p[61]          0.9999760
## p[62]          0.9999760
## p[63]          0.9999760
## p[64]          0.9998926
## p[65]          0.9998926
## p[66]          0.9999760
## p[67]          1.0001471
## p[68]          1.0001471
## p[69]          1.0001663
## p[70]          1.0001471
## p[71]          1.0001471
## p[72]          1.0001663
## p[73]          0.9998640
## p[74]          1.0000412
## p[75]          1.0000412
## p[76]          1.0000412
## p[77]          0.9998640
## p[78]          0.9998640
## p[79]          1.0000237
## p[80]          1.0000237
## p[81]          1.0000237
## p[82]          1.0000402
## p[83]          1.0000402
## p[84]          1.0000402
## p[85]          0.9996741
## p[86]          0.9996741
## p[87]          1.0000713
## p[88]          1.0000713
```

```
## p[89]         0.9996741
## p[90]         1.0000713
## p[91]         0.9999592
## p[92]         0.9999592
## p[93]         0.9998721
## p[94]         0.9999592
## p[95]         0.9998721
## p[96]         0.9998721
## p[97]         0.9997863
## p[98]         0.9998606
## p[99]         0.9997863
## p[100]        0.9998606
## p[101]        0.9997863
## p[102]        0.9998606
## p[103]        1.0000179
## p[104]        0.9999783
## p[105]        1.0000179
## p[106]        1.0000179
## p[107]        0.9999783
## p[108]        0.9999783
## p[109]        0.9999097
## p[110]        0.9999097
## p[111]        0.9999097
## p[112]        0.9998633
## p[113]        0.9999097
## p[114]        0.9999097
## p[115]        0.9999097
## p[116]        1.0000502
## p[117]        0.9998836
## p[118]        1.0000502
## p[119]        0.9998836
## p[120]        0.9998836
## p[121]        0.9998836
## p[122]        1.0000854
## p[123]        1.0000854
## p[124]        0.9998164
## p[125]        0.9998164
## p[126]        1.0000854
## p[127]        1.0000854
## p[128]        1.0000008
## p[129]        0.9999293
## p[130]        1.0000008
## p[131]        1.0000008
## p[132]        0.9999293
## p[133]        1.0000008
## p[134]        1.0000087
```

```
## p[135]        1.0000087
## p[136]        1.0000087
## p[137]        1.0000087
## p[138]        1.0003790
## p[139]        1.0000087
## p[140]        0.9999917
## p[141]        0.9998633
## p[142]        0.9999917
## p[143]        0.9998633
## p[144]        0.9999917
## p[145]        0.9996345
## p[146]        1.0000044
## p[147]        1.0000044
## p[148]        1.0000044
## p[149]        0.9996345
## p[150]        0.9996345
## p[151]        1.0001039
## p[152]        0.9995569
## p[153]        1.0001039
## p[154]        1.0001039
## p[155]        0.9995569
## p[156]        0.9995569
## p[157]        1.0000317
## p[158]        1.0001013
## p[159]        1.0001013
## p[160]        1.0001013
## p[161]        1.0000317
## p[162]        1.0000317
## p[163]        0.9999591
## p[164]        0.9998492
## p[165]        0.9998492
## p[166]        0.9998492
## p[167]        0.9999591
## p[168]        0.9999591
## p[169]        0.9997125
## p[170]        1.0002271
## p[171]        1.0002271
## p[172]        0.9997125
## p[173]        0.9997125
## p[174]        1.0002271
## p[175]        0.9998037
## p[176]        0.9998037
## p[177]        1.0002495
## p[178]        1.0002495
## p[179]        0.9998037
## p[180]        1.0002495
```

```
## p[181]        1.0002410
## p[182]        1.0000532
## p[183]        1.0002410
## p[184]        1.0000532
## p[185]        1.0002410
## p[186]        1.0000532
## p[187]        0.9997731
## p[188]        0.9997731
## p[189]        0.9997468
## p[190]        0.9997731
## p[191]        0.9997731
## p[192]        0.9997731
## p[193]        0.9997529
## p[194]        0.9998337
## p[195]        0.9997529
## p[196]        0.9998337
## p[197]        0.9998337
## p[198]        0.9997529
## p[199]        0.9998900
## p[200]        0.9998900
## p[201]        0.9998900
## p[202]        1.0002335
## p[203]        0.9998900
## p[204]        0.9998900
## p[205]        0.9998895
## p[206]        0.9998895
## p[207]        0.9998144
## p[208]        0.9998895
## p[209]        0.9998895
## p[210]        0.9998895
## p[211]        1.0000680
## p[212]        1.0000680
## p[213]        1.0000680
## p[214]        0.9999348
## p[215]        1.0000680
## p[216]        1.0000680
## p[217]        0.9996543
## p[218]        0.9998186
## p[219]        0.9996543
## p[220]        0.9996543
## p[221]        0.9998186
## p[222]        0.9998186
## p[223]        0.9997049
## p[224]        0.9999011
## p[225]        0.9997049
## p[226]        0.9999011
```

```
## p[227]        0.9997049
## p[228]        0.9999011
## p[229]        0.9998952
## p[230]        1.0000069
## p[231]        0.9998952
## p[232]        0.9998952
## p[233]        1.0000069
## p[234]        1.0000069
## p[235]        0.9998291
## p[236]        0.9998151
## p[237]        0.9998291
## p[238]        0.9998151
## p[239]        0.9998291
## p[240]        0.9998151
## p[241]        1.0001217
## p[242]        1.0001217
## p[243]        0.9999027
## p[244]        0.9999027
## p[245]        1.0001217
## p[246]        0.9999027
## p[247]        0.9999783
## p[248]        0.9999783
## p[249]        1.0001466
## p[250]        1.0001466
## p[251]        0.9999783
## p[252]        1.0001466
## p[253]        1.0001377
## p[254]        1.0000077
## p[255]        1.0001377
## p[256]        1.0000077
## p[257]        1.0000077
## p[258]        1.0000077
## p[259]        0.9997683
## p[260]        0.9997683
## p[261]        0.9999454
## p[262]        0.9997683
## p[263]        0.9997683
## p[264]        0.9997683
## p[265]        0.9998197
## p[266]        0.9998197
## p[267]        0.9997561
## p[268]        0.9998197
## p[269]        0.9997561
## p[270]        0.9997561
## p[271]        0.9999994
## p[272]        0.9999994
```

```
## p[273]        0.9999994
## p[274]        0.9999994
## p[275]        1.0001193
## p[276]        0.9999994
## p[277]        0.9996592
## p[278]        1.0001386
## p[279]        0.9996592
## p[280]        0.9996592
## p[281]        0.9996592
## p[282]        0.9996592
## p[283]        1.0004651
## p[284]        1.0003025
## p[285]        1.0003025
## p[286]        1.0004651
## p[287]        1.0003025
## p[288]        1.0003025
## p[289]        1.0002098
## p[290]        0.9999281
## p[291]        0.9999281
## p[292]        0.9999281
## p[293]        1.0002098
## p[294]        1.0002098
## p[295]        0.9998923
## p[296]        0.9998923
## p[297]        0.9999175
## p[298]        0.9999175
## p[299]        0.9998923
## p[300]        0.9999175
## p[301]        0.9995971
## p[302]        0.9995971
## p[303]        1.0002573
## p[304]        0.9995971
## p[305]        1.0002573
## p[306]        1.0002573
## p[307]        0.9998781
## p[308]        0.9997288
## p[309]        0.9998781
## p[310]        0.9998781
## p[311]        0.9997288
## p[312]        0.9997288
## p[313]        1.0007034
## p[314]        1.0007034
## p[315]        1.0007034
## p[316]        0.9998567
## p[317]        0.9998567
## p[318]        0.9998567
```

```
## p[319]        1.0005774
## p[320]        1.0005774
## p[321]        1.0002840
## p[322]        1.0005774
## p[323]        1.0002840
## p[324]        1.0002840
## p[325]        0.9998759
## p[326]        0.9997855
## p[327]        0.9998759
## p[328]        0.9997855
## p[329]        0.9997855
## p[330]        0.9998759
## p[331]        0.9998335
## p[332]        1.0002206
## p[333]        0.9998335
## p[334]        1.0002206
## p[335]        0.9998335
## p[336]        1.0002206
## p[337]        0.9999903
## p[338]        1.0000378
## p[339]        1.0000378
## p[340]        0.9999903
## p[341]        0.9999903
## p[342]        1.0000378
## p[343]        0.9998525
## p[344]        0.9998525
## p[345]        0.9998600
## p[346]        0.9998600
## p[347]        0.9998600
## p[348]        0.9998525
## p[349]        0.9999039
## p[350]        0.9997538
## p[351]        0.9997538
## p[352]        0.9999039
## p[353]        0.9999039
## p[354]        0.9999039
## p[355]        1.0005274
## p[356]        1.0005274
## p[357]        1.0001710
## p[358]        1.0001710
## p[359]        1.0005274
## p[360]        1.0005274
## p[361]        0.9996811
## p[362]        0.9999244
## p[363]        0.9999244
## p[364]        0.9996811
```

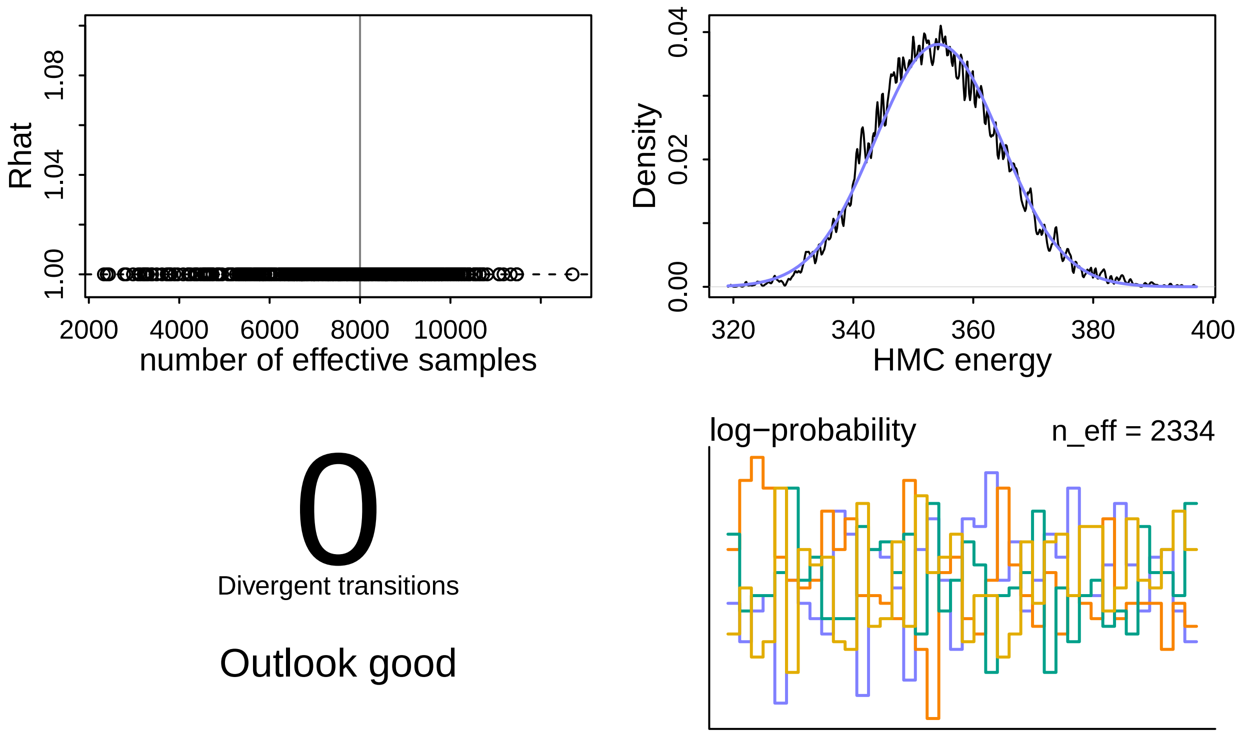
```
## p[365]        0.9999244
## p[366]        0.9996811
## p[367]        0.9996774
## p[368]        0.9996774
## p[369]        0.9996774
## p[370]        0.9999252
## p[371]        0.9999252
## p[372]        0.9999252
## p[373]        1.0001409
## p[374]        1.0001409
## p[375]        1.0001409
## p[376]        0.9998408
## p[377]        0.9998408
## p[378]        0.9998408
## p[379]        0.9996134
## p[380]        0.9996690
## p[381]        0.9996134
## p[382]        0.9996690
## p[383]        0.9996134
## p[384]        0.9996690
## p[385]        0.9998375
## p[386]        1.0000138
## p[387]        1.0000138
## p[388]        0.9998375
## p[389]        1.0000138
## p[390]        0.9998375
## p[391]        1.0003768
## p[392]        1.0003768
## p[393]        1.0000894
## p[394]        1.0000894
## p[395]        1.0000894
## p[396]        1.0003768
## p[397]        1.0001648
## p[398]        1.0000542
## p[399]        1.0001648
## p[400]        1.0001648
## p[401]        1.0000542
## p[402]        1.0000542
## p[403]        0.9997514
## p[404]        0.9997514
## p[405]        0.9999767
## p[406]        0.9999767
## p[407]        0.9999767
## p[408]        0.9999767
## p[409]        1.0002691
## p[410]        1.0002691
```

```
## p[411]        0.9995635
## p[412]        1.0002691
## p[413]        0.9995635
## p[414]        0.9995635
## p[415]        0.9998917
## p[416]        0.9998917
## p[417]        0.9999167
## p[418]        0.9999167
## p[419]        0.9998917
## p[420]        0.9999167
## p[421]        1.0001678
## p[422]        1.0001678
## p[423]        0.9999359
## p[424]        0.9999359
## p[425]        0.9999359
## p[426]        1.0001678
## p[427]        0.9999155
## p[428]        0.9999155
## p[429]        1.0004081
## p[430]        0.9999155
## p[431]        1.0004081
## p[432]        0.9999155
## p[433]        0.9996120
## p[434]        1.0000495
## p[435]        1.0000495
## p[436]        0.9996120
## p[437]        0.9996120
## p[438]        1.0000495
## p[439]        1.0000025
## p[440]        1.0000025
## p[441]        0.9996947
## p[442]        1.0000025
## p[443]        0.9996947
## p[444]        0.9996947
## p[445]        0.9997684
## p[446]        0.9997684
## p[447]        1.0004120
## p[448]        1.0004120
## p[449]        1.0004120
## p[450]        0.9997684
## p[451]        1.0001953
## p[452]        0.9996114
## p[453]        1.0001953
## p[454]        1.0001953
## p[455]        0.9996114
## p[456]        0.9996114
```

```
## p[457]        0.9999909
## p[458]        1.0000336
## p[459]        1.0000336
## p[460]        1.0000336
## p[461]        0.9999909
## p[462]        0.9999909
## p[463]        0.9999604
## p[464]        0.9997714
## p[465]        0.9997714
## p[466]        0.9997714
## p[467]        0.9999604
## p[468]        0.9999604
## p[469]        1.0000097
## p[470]        0.9998892
## p[471]        1.0000097
## p[472]        0.9998892
## p[473]        1.0000097
## p[474]        1.0000097
## p[475]        0.9997711
## p[476]        0.9998346
## p[477]        0.9997711
## p[478]        0.9998346
## p[479]        0.9997711
## p[480]        0.9998346
## p[481]        0.9999005
## p[482]        0.9997826
## p[483]        0.9997826
## p[484]        0.9999005
## p[485]        0.9999005
## p[486]        0.9997826
## p[487]        0.9997496
## p[488]        0.9997496
## p[489]        1.0000738
## p[490]        0.9997496
## p[491]        0.9997496
## p[492]        0.9997496
## p[493]        0.9997402
## p[494]        0.9998467
## p[495]        0.9998467
## p[496]        0.9997402
## p[497]        0.9997402
## p[498]        0.9997402
## p[499]        1.0001603
## p[500]        1.0001603
## p[501]        1.0001603
## p[502]        0.9998360
```

```
## p[503]        0.9998360
## p[504]        0.9998360
## Rho_A[1,1]          NaN
## Rho_A[1,2]    0.9998256
## Rho_A[1,3]    0.9999680
## Rho_A[1,4]    0.9999356
## Rho_A[2,1]    0.9998256
## Rho_A[2,2]          NaN
## Rho_A[2,3]    0.9997584
## Rho_A[2,4]    0.9997296
## Rho_A[3,1]    0.9999680
## Rho_A[3,2]    0.9997584
## Rho_A[3,3]          NaN
## Rho_A[3,4]    0.9998517
## Rho_A[4,1]    0.9999356
## Rho_A[4,2]    0.9997296
## Rho_A[4,3]    0.9998517
## Rho_A[4,4]          NaN
## Rho_B[1,1]          NaN
## Rho_B[1,2]    1.0004023
## Rho_B[1,3]    0.9999085
## Rho_B[1,4]    0.9998687
## Rho_B[2,1]    1.0004023
## Rho_B[2,2]          NaN
## Rho_B[2,3]    0.9998125
## Rho_B[2,4]    1.0001271
## Rho_B[3,1]    0.9999085
## Rho_B[3,2]    0.9998125
## Rho_B[3,3]          NaN
## Rho_B[3,4]    1.0004270
## Rho_B[4,1]    0.9998687
## Rho_B[4,2]    1.0001271
## Rho_B[4,3]    1.0004270
## Rho_B[4,4]          NaN
```

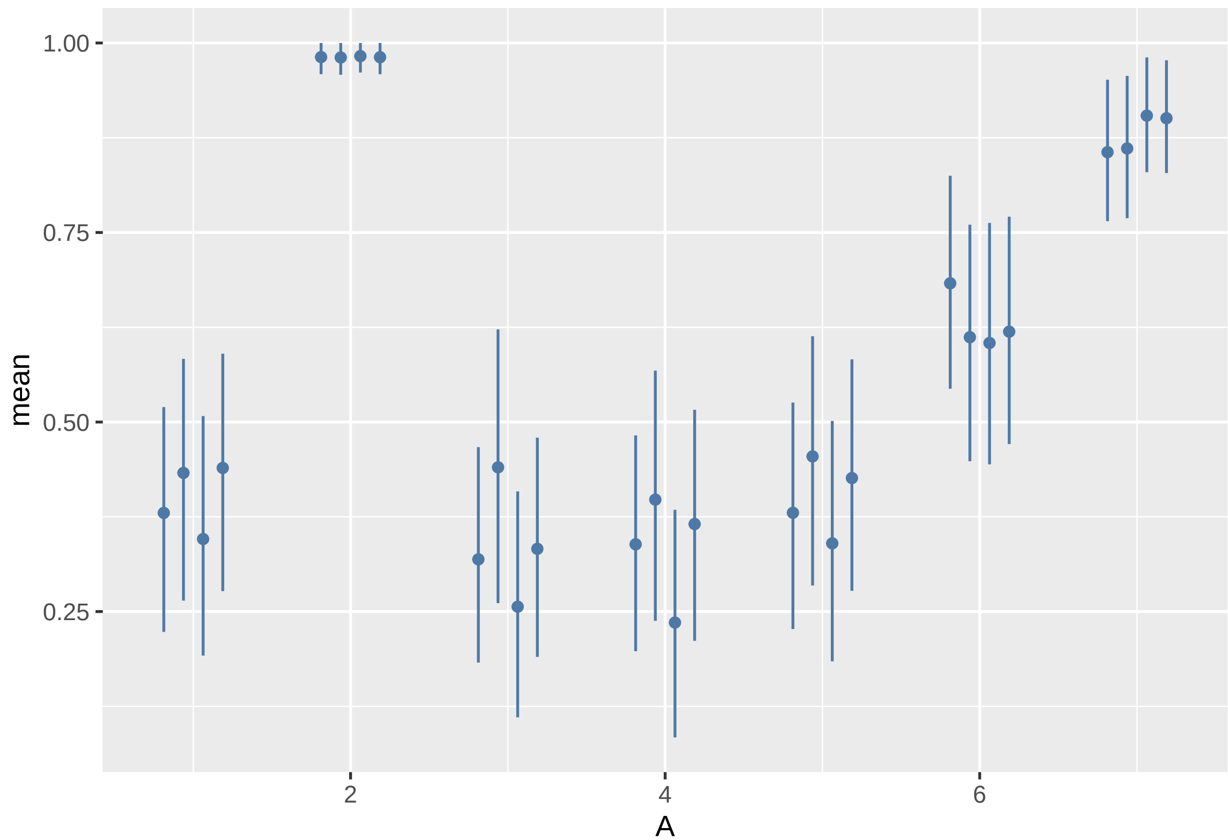
```r
dashboard(m1)
```




```
post <- extract.samples(m1)

araw <- inv_logit(apply(post$a, 3, function(x) x + post$abar))
pA <- cbind(expand.grid(S=1:dim(post$a)[1], A=1:dim(post$a)[2], T=c('R/N','L/N','R/P','L/P
  group_by(A,T) %>%
  summarise(mean=mean(p), hpdi=HPDI(p)) %>%
  ungroup() %>%
  group_by(A,T,mean) %>%
    summarise(hpdi_lower=min(hpdi), hpdi_upper=max(hpdi)) %>%
  as.data.frame() %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'A', 'T'. You can override using the `.groups` argum
## `summarise()` has grouped output by 'A', 'T'. You can override using the `.groups` argum
```

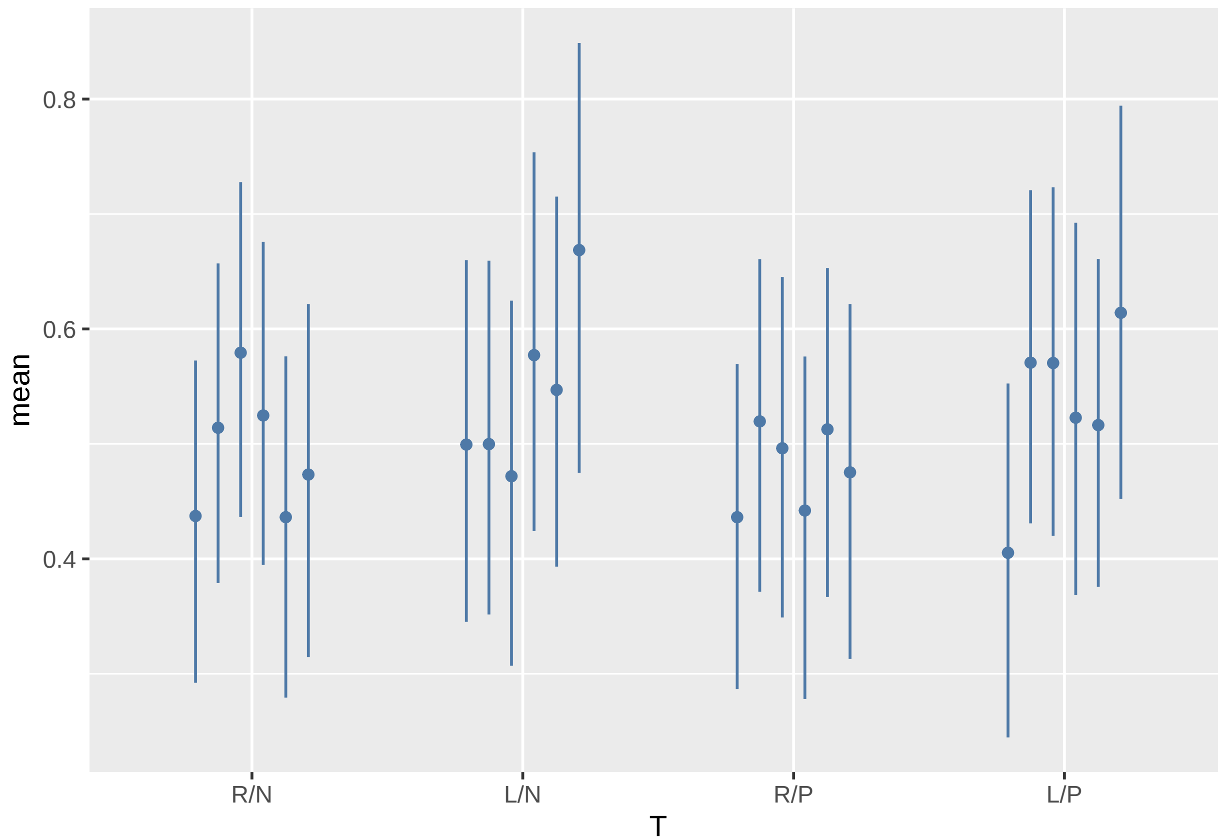
```
ggplot(pA) +
    geom_point(aes(x=A, y=mean, colour='', group=`T`), position=position_dodge(width=0.5))
    geom_linerange(aes(x=A, ymin=hpdi_lower, ymax=hpdi_upper, colour='', group=`T`), positi
    theme(legend.position='none') +
    scale_color_tableau()
```



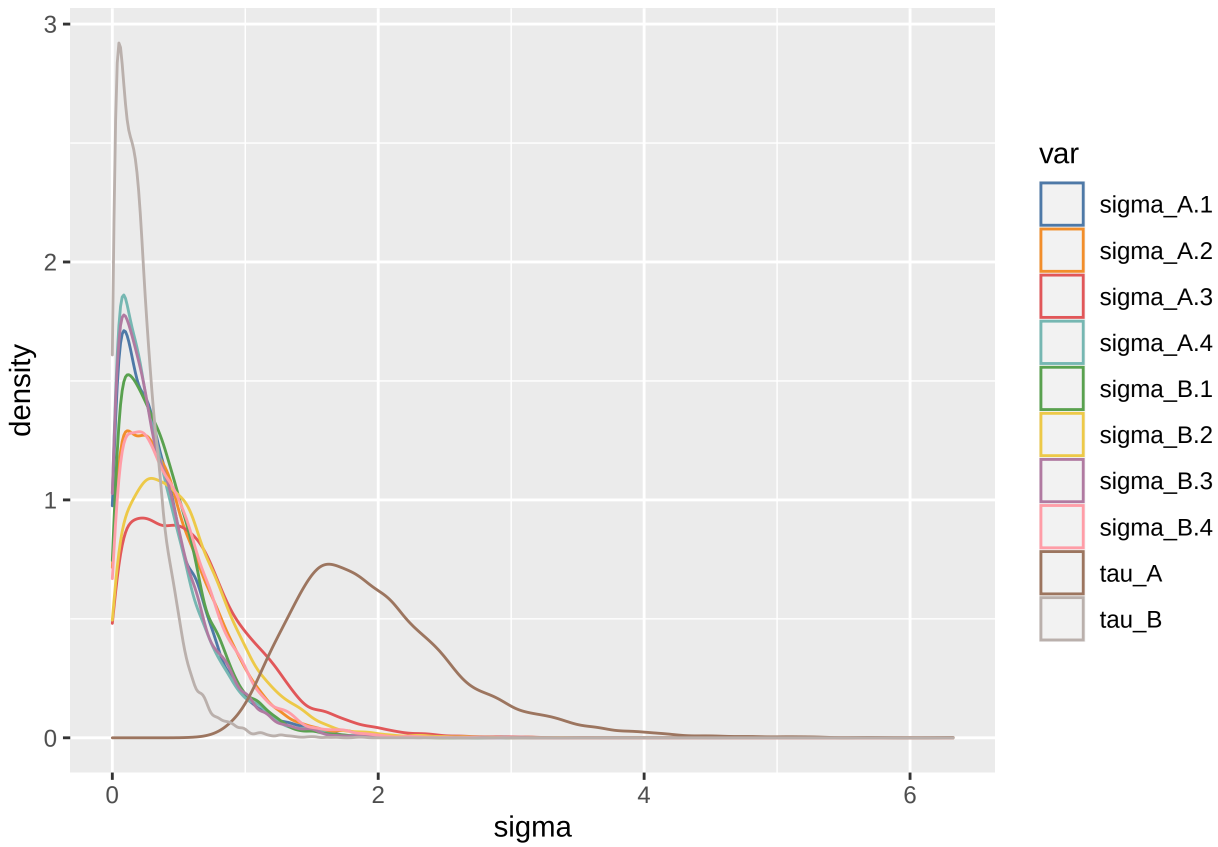

```
braw <- inv_logit(apply(post$b, 3, function(x) x + post$bbar))
pB <- cbind(expand.grid(S=1:dim(post$b)[1], B=1:dim(post$b)[2], T=c('R/N','L/N','R/P','L/P
    select(-S) %>%
    group_by(B,T) %>%
    summarise(mean=mean(p), hpdi=HPDI(p)) %>%
  ungroup() %>%
  group_by(B,T,mean) %>%
    summarise( hpdi_lower=min(hpdi), hpdi_upper=max(hpdi)) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'B', 'T'. You can override using the `.groups` argu
## `summarise()` has grouped output by 'B', 'T'. You can override using the `.groups` argu
```

```
ggplot(pB) +
    geom_point(aes(x=T, y=mean, colour='', group=B), position=position_dodge(width=0.5)) +
    geom_linerange(aes(x=T, ymin=hpdi_lower, ymax=hpdi_upper, colour='', group=B), positio
    theme(legend.position='none')+
    scale_color_tableau()
```




```
data.frame(sigma_A=post$sigma_A, sigma_B=post$sigma_B, tau_A=post$tau_A, tau_B=post$tau_B)
    pivot_longer(everything(), names_to='var', values_to='sigma') %>%
    ggplot() +
    geom_density(aes(x=sigma, colour=var)) +
    scale_colour_tableau()
```



Lecture 15: Social Networks

Networks

Mainly focused on Dyads, pair-wise relationships.

What we are interested in is giving relationships, how much is reciprocal.

$$\begin{tikzcd}
& TBA \arrow[d] & \\
HA \arrow[ur] \arrow[r] \arrow[dr] & GAB & HB \arrow[ul] \arrow[l] \arrow[dl] \\
& TAB \arrow[u] &
\end{tikzcd}$$

Social giving G_{AB} between households H_A to H_B based on one view of tie to the other T_{AB}. Note that T_{AB} can not be observed, we really need to work with a generative model to get understanding about how we infer it.

Adhockery

Permutation of network structure does not give a null model.

Social Network Model

$$
\begin{aligned}
G_{AB} &\sim \text{Poisson}(\lambda_{AB}) \\
\log(\lambda_{AB}) &= \alpha + T_{AB} \\
G_{BA} &\sim \text{Poisson}(\lambda_{BA}) \\
\log(\lambda_{BA}) &= \alpha + T_{BA} \\
\begin{pmatrix} T_{AB} \\ T_{BA} \end{pmatrix} &\sim \text{MVNormal}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{bmatrix} \sigma^2 & \rho\sigma^2 \\ \rho\sigma^2 & \sigma \end{bmatrix} \right) \\
\alpha &\sim \text{Normal}(0, 1) \\
\sigma &\sim \text{Exponential}(1) \\
\rho &\sim \text{LJKCorr}(2)
\end{aligned}
$$

Note that because we have dyads as our model we need two sets, one for each direction. (There are twice as many Ts and Gs as dyads).

This model does not include the confounding household characteristics in our DAG so we need to introduce generalised giving G and receiving R.

$$
\begin{aligned}
G_{AB} &\sim \text{Poisson}(\lambda_{AB}) \\
\log(\lambda_{AB}) &= \alpha + T_{AB} + G_A + R_B \\
G_{BA} &\sim \text{Poisson}(\lambda_{BA}) \\
\log(\lambda_{BA}) &= \alpha + T_{BA} + G_B + R_A \\
\begin{pmatrix} T_{AB} \\ T_{BA} \end{pmatrix} &\sim \text{MVNormal}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma^2 & \rho\sigma^2 \\ \rho\sigma^2 & \sigma^2 \end{bmatrix} \right) \\
\begin{pmatrix} G_A \\ R_A \end{pmatrix} &\sim \text{MVNormal}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, R_{GR}, S_{GR} \right) \\
\alpha &\sim \text{Normal}(0, 1) \\
\sigma, S_{GR} &\sim \text{Exponential}(1) \\
\rho, R_{GR} &\sim \text{LJKCorr}(2)
\end{aligned}
$$

Posterior Social Networks

Now the resultant network does not give just one network, some of the network might be stable between samples of the posterior, but others will change. The inference made on the network downstream must take multiple samples from the network posterior for the calculation, giving you the inherited uncertainty.

Also remember that there are relationships beyond two, so tryads and onwards could be important.

Association Index

What if we also had a measure of association A_{AB} between two households and their wealth W_A.

$$\begin{aligned}
G_{AB} &\sim \text{Poisson}(\lambda_{AB}) \\
\log(\lambda_{AB}) &= \alpha + T_{AB} + \beta_A A_{AB} + G_A + \beta_{WG} W_A + R_B + \beta_{WR} W_B \\
G_{BA} &\sim \text{Poisson}(\lambda_{BA}) \\
\log(\lambda_{BA}) &= \alpha + T_{BA} + \beta_B A_{AB} + G_B + \beta_{WG} W_B + R_A + \beta_{WR} W_A \\
\begin{pmatrix} T_{AB} \\ T_{BA} \end{pmatrix} &\sim \text{MVNormal}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma^2 & \rho\sigma^2 \\ \rho\sigma^2 & \sigma^2 \end{bmatrix} \right) \\
\begin{pmatrix} G_A \\ R_A \end{pmatrix} &\sim \text{MVNormal}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, R_{GR}, S_{GR} \right) \\
\alpha, \beta_{j,R}, \beta_{j,G}, \beta_A &\sim \text{Normal}(0,1) \\
\sigma, S_{GR} &\sim \text{Exponential}(1) \\
\rho, R_{GR} &\sim \text{LJKCorr}(2)
\end{aligned}$$

Lecture 16: Gaussian Process

Kernal Functions

Rather than using a correlation matrix to determine covariation uses a functional form, a kernal function. This allows infinite dimensional normals, but in a sense with even less parameters than correlation approaches. One does have to pick the correct kernal function. This just replaces the correlation matrix from before.

Quadratic (L2)

$$k(x,y) = \alpha^2 \exp\left(-\frac{(x-y)^2}{\sigma^2}\right)$$

Ornstein-Uhlenbeck (L1)

$$k(x,y) = \alpha^2 \exp\left(-\frac{|x-y|}{\sigma}\right)$$

Periodic

$$k(x,y) = \alpha^2 \exp\left(-\frac{2\sin^2((x-y)/2)}{\sigma^2}\right)$$

Phylogony

Remember phylogeny doesn't exist, it is just a potentially useful model with some features we want. In fact there are multiple phylogenies for each dataset that could be argued for. Ideally we want to fit our model over the phylogeny at the same time we fit our phylogeny and then make inferences drawing from the posterior from this.

Further Gaussian Progression

- **Automatic relevance determination (ARD):** Automatic weight fitting across multiple parameters, figuring out automatically relative weights in your model. Used a lot in machine learning.
- **Multi Output Gaussian Proccess:** Rather than outputting a single value from distance, output a vector.
- **Kalman Filters:** Noisey instrumentation.

Lecture 17: Measurement Error

Resist the urge to be clever

Error's in DAGs

$$Ptrue \longleftarrow Pmeas \longleftarrow eP$$

Marriage Rates with Errors

We had marriage rate happines model, with age being a factor.

$$\begin{tikzcd}
 & A \arrow[dl] \arrow[dr] & \\
M \arrow[rr] & & H
\end{tikzcd}$$

Now imagine we had error of measurement in all of these

$$\begin{tikzcd}
 & & eA \arrow[d] & & \\
 & & Ao & & \\
eM \arrow[d] & & At \arrow[u] \arrow[dl] \arrow[dr] & & eH \arrow[d] \\
Mo & Mt \arrow[l] \arrow[rr] & & Ht \arrow[r] & Ho
\end{tikzcd}$$

Noting that we no longer observe the true values in our relationship. In fact we can now think of confounding variables that affect the error, for instance a population. However lets walk back a step and just go with this model. Let's start with just D, we use two simultaneous models

$$
\begin{aligned}
D_i^{true} &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_M M_i + \beta_A A_i \\
\\
D_i^{obs} &\sim \text{Normal}(D_i^{true}, S_i)
\end{aligned}
$$

So lets run this model, and see what the results does.

```
data(WaffleDivorce)
df <- WaffleDivorce

d <- list(
    D_obs = standardize( df$Divorce )
,   D_std = df$Divorce.SE / sd(df$Divorce)
,   M     = standardize( df$Marriage )
,   A     = standardize( df$MedianAgeMarriage )
,   N     = nrow(df)
)
```

Then it is a simple as writing up the dual models. Yes this is exactly a partial pooling.

```
m0 <- cstan(file='../models/l17_m0.stan', data=d, chains=4, cores=4, threads=2, iter=4000)
```

```
## Warning in readLines(stan_file): incomplete final line found on '../models/
## l17_m0.stan'

## Running MCMC with 4 parallel chains, with 2 thread(s) per chain...

## Chain 1 Rejecting initial value:

## Chain 1   Error evaluating the log probability at the initial value.

## Chain 1 Exception: exponential_lpdf: Random variable is -0.675071, but must be nonnegati
## Chain 1 Exception: exponential_lpdf: Random variable is -0.675071, but must be nonnegati

## Chain 1 Rejecting initial value:

## Chain 1   Error evaluating the log probability at the initial value.

## Chain 1 Exception: exponential_lpdf: Random variable is -1.92603, but must be nonnegativ
## Chain 1 Exception: exponential_lpdf: Random variable is -1.92603, but must be nonnegativ

## Chain 1 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 1 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 1 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 1 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 1 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 1 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 1 Iteration:  600 / 4000 [ 15%]  (Warmup)

## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b
```

```
## Chain 1 Exception: exponential_lpdf: Random variable is -841.327, but must be nonnegativ
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 1 but if this warning occurs often then your model may be either severely ill-conc
## Chain 1
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 1 Exception: exponential_lpdf: Random variable is -6.27816, but must be nonnegativ
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 1 but if this warning occurs often then your model may be either severely ill-conc
## Chain 1
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 1 Exception: exponential_lpdf: Random variable is -1.95192, but must be nonnegativ
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 1 but if this warning occurs often then your model may be either severely ill-conc
## Chain 1
## Chain 2 Rejecting initial value:
## Chain 2   Error evaluating the log probability at the initial value.
## Chain 2 Exception: exponential_lpdf: Random variable is -0.553784, but must be nonnegati
## Chain 2 Exception: exponential_lpdf: Random variable is -0.553784, but must be nonnegati
## Chain 2 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 2 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 2 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 2 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 2 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 2 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 2 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 2 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 2 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 2 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 3 Rejecting initial value:
## Chain 3   Error evaluating the log probability at the initial value.
## Chain 3 Exception: exponential_lpdf: Random variable is -1.29969, but must be nonnegativ
## Chain 3 Exception: exponential_lpdf: Random variable is -1.29969, but must be nonnegativ
## Chain 3 Rejecting initial value:
## Chain 3   Error evaluating the log probability at the initial value.
```

```
## Chain 3 Exception: exponential_lpdf: Random variable is -1.77823, but must be nonnegativ
## Chain 3 Exception: exponential_lpdf: Random variable is -1.77823, but must be nonnegativ
## Chain 3 Rejecting initial value:
## Chain 3   Error evaluating the log probability at the initial value.
## Chain 3 Exception: exponential_lpdf: Random variable is -1.18041, but must be nonnegativ
## Chain 3 Exception: exponential_lpdf: Random variable is -1.18041, but must be nonnegativ
## Chain 3 Rejecting initial value:
## Chain 3   Error evaluating the log probability at the initial value.
## Chain 3 Exception: exponential_lpdf: Random variable is -1.80749, but must be nonnegativ
## Chain 3 Exception: exponential_lpdf: Random variable is -1.80749, but must be nonnegativ
## Chain 3 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 3 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 3 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 3 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 3 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 3 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 3 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 3 Exception: exponential_lpdf: Random variable is -96.711, but must be nonnegative
## Chain 3 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 3 but if this warning occurs often then your model may be either severely ill-cond
## Chain 3
## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 3 Exception: exponential_lpdf: Random variable is -0.102028, but must be nonnegati
## Chain 3 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 3 but if this warning occurs often then your model may be either severely ill-cond
## Chain 3
## Chain 4 Rejecting initial value:
## Chain 4   Error evaluating the log probability at the initial value.
## Chain 4 Exception: exponential_lpdf: Random variable is -0.659811, but must be nonnegati
## Chain 4 Exception: exponential_lpdf: Random variable is -0.659811, but must be nonnegati
## Chain 4 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 4 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 4 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 4 Iteration:  300 / 4000 [  7%]  (Warmup)
```

```
## Chain 4 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 4 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 4 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 4 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 4 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 4 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 4 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 4 Iteration: 1100 / 4000 [ 27%]  (Warmup)

## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 4 Exception: exponential_lpdf: Random variable is -1067.21, but must be nonnegativ

## Chain 4 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 4 but if this warning occurs often then your model may be either severely ill-conc

## Chain 4

## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 4 Exception: exponential_lpdf: Random variable is -10.5759, but must be nonnegativ

## Chain 4 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 4 but if this warning occurs often then your model may be either severely ill-conc

## Chain 4

## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 4 Exception: exponential_lpdf: Random variable is -0.0207578, but must be nonnegat

## Chain 4 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 4 but if this warning occurs often then your model may be either severely ill-conc

## Chain 4

## Chain 1 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 1 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 1 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 1 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 1 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 1 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 1 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 1 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 2 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 2 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 2 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 2 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 2 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 2 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 2 Iteration: 1600 / 4000 [ 40%]  (Warmup)
```

```
## Chain 2 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 2 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 2 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 2 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 2 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 2 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 2 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 3 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 3 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 3 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 3 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 3 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 3 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 3 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 3 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 4 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 4 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 4 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 4 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 4 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 4 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 4 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 4 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 4 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 4 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 4 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 4 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 4 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 4 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 1 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 1 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 1 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 1 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 1 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 1 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 2 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 2 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 2 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 2 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 2 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 2 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 2 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 2 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 2 Iteration: 3100 / 4000 [ 77%]  (Sampling)
```

```
## Chain 2 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 2 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 2 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 3 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 3 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 3 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 3 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 3 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 3 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 3 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 3 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 4 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 4 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 4 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 4 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 4 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 4 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 4 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 4 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 4 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 4 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 4 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 4 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 4 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 1 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 1 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 1 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 1 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 1 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 2 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 2 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 2 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 2 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 2 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 2 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 3 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 3 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 3 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 3 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 3 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 3 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 3 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 4 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 4 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 4 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 2 finished in 0.4 seconds.
```

```
## Chain 4 finished in 0.4 seconds.
## Chain 1 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 1 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 1 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 1 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 1 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 1 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 3 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 3 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 3 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 3 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 3 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 3 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 3 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 3 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 1 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 1 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 1 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 1 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 1 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 1 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 1 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 1 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 3 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 3 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 3 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 3 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 1 finished in 0.6 seconds.
## Chain 3 finished in 0.6 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.5 seconds.
## Total execution time: 0.7 seconds.
```

```
precis(m0, depth=2)
```

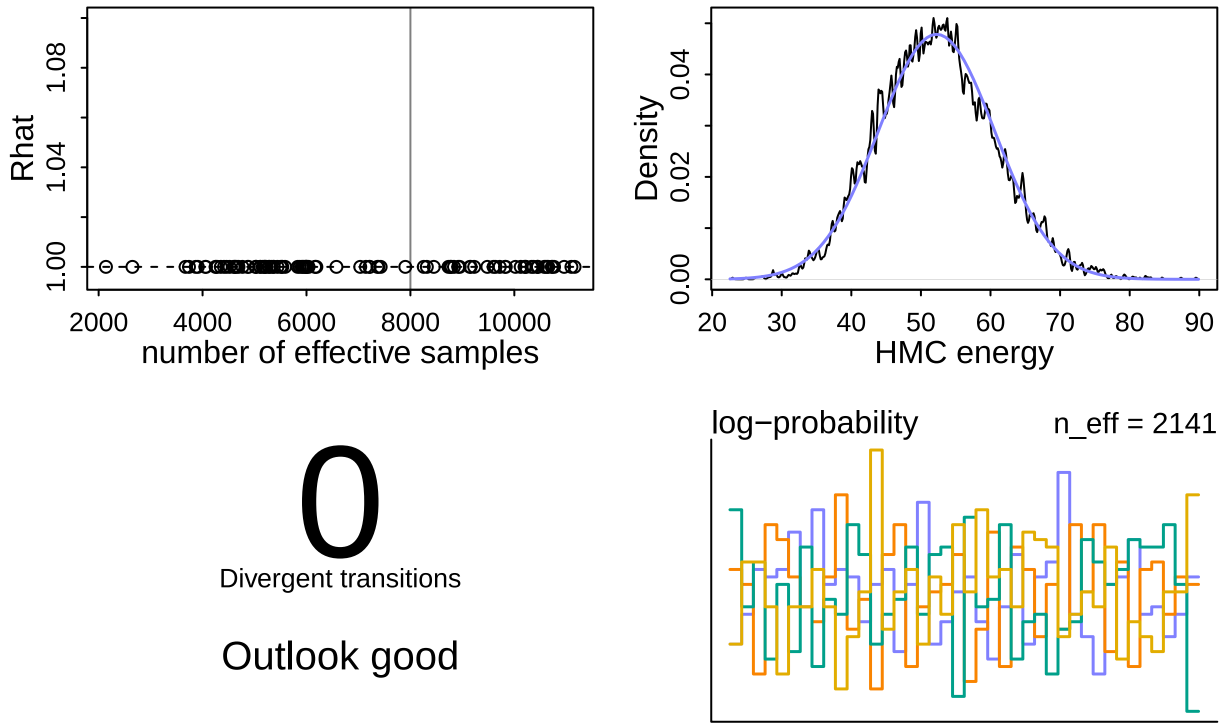
```
##                  mean         sd         5.5%        94.5%     n_eff     Rhat4
## D_true[1]   1.16742871 0.36221169  0.595105960  1.756526500  7216.723 1.0000700
## D_true[2]   0.69229966 0.54712316 -0.169757500  1.580815000  7383.034 1.0003142
## D_true[3]   0.43239099 0.33551534 -0.101465125  0.965043445 10442.432 0.9998115
## D_true[4]   1.41689156 0.45882940  0.698220665  2.152825550  8733.576 0.9998453
## D_true[5]  -0.90195255 0.12737816 -1.104142750 -0.695369130 10212.789 0.9995894
## D_true[6]   0.65440945 0.39939570  0.025491608  1.301362200  8315.989 0.9997394
## D_true[7]  -1.36670051 0.34627777 -1.921012000 -0.813088460  9144.853 0.9995681
## D_true[8]  -0.32987795 0.48218107 -1.092762100  0.441351370  9719.485 1.0002108
## D_true[9]  -1.86666884 0.60331166 -2.803117600 -0.901699550  7034.597 1.0001952
```

```
## D_true[10] -0.61975156 0.16693750 -0.885186760 -0.354681195  9821.117 1.0001124
## D_true[11]  0.76761370 0.28806640  0.311660170  1.235172750  7902.194 1.0000238
## D_true[12] -0.53947363 0.47676369 -1.296287500  0.216294675  7212.838 1.0001265
## D_true[13]  0.18049427 0.50306316 -0.644381480  0.965500540  5158.204 0.9997396
## D_true[14] -0.86547719 0.23363418 -1.238735750 -0.497475945 10448.616 1.0000459
## D_true[15]  0.55582045 0.29044416  0.105514325  1.018469800  9835.311 1.0000179
## D_true[16]  0.27601486 0.38712393 -0.352667780  0.883680550 10225.557 0.9999594
## D_true[17]  0.49381739 0.42405153 -0.169325880  1.170001100 10454.103 1.0001141
## D_true[18]  1.25241304 0.35719444  0.692079985  1.827735400  8449.849 0.9998909
## D_true[19]  0.43667480 0.37983005 -0.164177765  1.052562200  8776.308 0.9996805
## D_true[20]  0.39379509 0.55211807 -0.454756830  1.289819300  5588.203 0.9999344
## D_true[21] -0.55484372 0.32003198 -1.065138750 -0.044668699  9636.801 1.0004112
## D_true[22] -1.09963554 0.26217434 -1.522209800 -0.679600960  8919.941 0.9997026
## D_true[23] -0.27243971 0.26297960 -0.697581670  0.144151720 11098.822 0.9996945
## D_true[24] -1.00341168 0.29653450 -1.480137700 -0.531820725  8792.180 0.9996526
## D_true[25]  0.42520851 0.41615493 -0.236362305  1.094683850 10022.041 0.9997369
## D_true[26] -0.03551918 0.30348356 -0.519090925  0.444836765 11159.166 0.9999780
## D_true[27] -0.03154208 0.50697092 -0.837001970  0.781541725  9481.516 1.0001179
## D_true[28] -0.15872267 0.38994906 -0.785314820  0.457162310 10196.504 0.9997119
## D_true[29] -0.26507405 0.50932146 -1.069643850  0.554464595  8257.334 0.9999297
## D_true[30] -1.80090878 0.23767969 -2.183835750 -1.424844150  9600.666 0.9996823
## D_true[31]  0.16927274 0.42861885 -0.503238455  0.868017825 10602.605 0.9996316
## D_true[32] -1.65996382 0.16413331 -1.921983300 -1.400545600 10577.458 1.0001646
## D_true[33]  0.11534041 0.24058709 -0.273061510  0.498928520 10730.517 0.9999210
## D_true[34] -0.05154229 0.50968560 -0.876331370  0.743704225  7365.627 0.9998685
## D_true[35] -0.12609448 0.22882170 -0.491780280  0.241820910 10652.781 0.9996564
## D_true[36]  1.27254976 0.42526687  0.593887110  1.955964300  9160.000 0.9998671
## D_true[37]  0.23143042 0.35502461 -0.328439835  0.806965440 10959.126 0.9997302
## D_true[38] -1.02777265 0.21920849 -1.378726950 -0.676353120 10632.920 0.9995646
## D_true[39] -0.92213828 0.52762822 -1.748369250 -0.078246318  7427.664 0.9997716
## D_true[40] -0.67471516 0.32194059 -1.190134400 -0.162727605 10331.172 1.0000699
## D_true[41]  0.24534453 0.54302744 -0.615620110  1.106519900  9227.947 0.9999332
## D_true[42]  0.73949856 0.33922109  0.196831975  1.284287500  8768.205 0.9998023
## D_true[43]  0.19278206 0.18354491 -0.097223061  0.483195310 10764.233 0.9996798
## D_true[44]  0.80080684 0.41823084  0.119651775  1.456278250  6576.093 0.9999460
## D_true[45] -0.40942766 0.51964166 -1.220691000  0.413615025  8839.377 0.9998764
## D_true[46] -0.38555594 0.25382866 -0.788167265  0.025921732 10722.307 1.0001572
## D_true[47]  0.13561094 0.30232186 -0.348468105  0.614583445 10122.530 0.9999113
## D_true[48]  0.55430590 0.47173490 -0.201010235  1.309953450 10370.117 0.9996636
## D_true[49] -0.63424091 0.27883856 -1.083107950 -0.191830175  8948.016 1.0004487
## D_true[50]  0.86003486 0.58510292 -0.109249895  1.749547050  7143.541 1.0000623
## alpha      -0.05460165 0.09490717 -0.204300770  0.098466665  5984.950 1.0001925
## beta_A     -0.60791630 0.15902270 -0.855552750 -0.350585555  4257.658 1.0005409
## beta_M      0.06041299 0.16514710 -0.203074605  0.326244515  3673.036 1.0011103
## sigma       0.58548999 0.10671578  0.425432015  0.764086585  2650.093 1.0006828
## mu[1]       0.31533962 0.12968932  0.107592280  0.521970540  5534.638 0.9996849
```

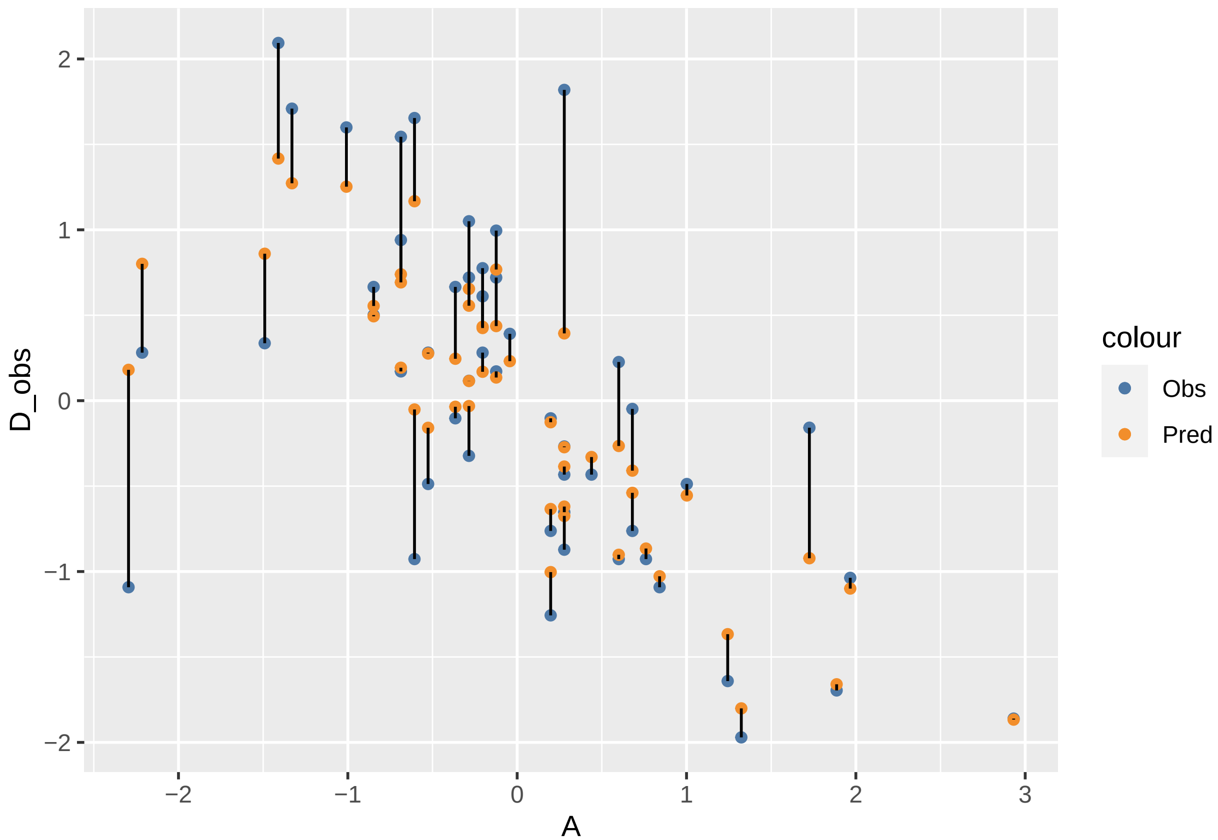
```
## mu[2]        0.45648217 0.22987422  0.089166131  0.823750175  4250.266 1.0009541
## mu[3]        0.07251832 0.09789652 -0.083392100  0.229653960  6189.791 0.9998714
## mu[4]        0.90278649 0.23246990  0.525756980  1.263320550  5187.688 1.0003428
## mu[5]       -0.43539395 0.12002785 -0.621193335 -0.238767545  5228.117 1.0006881
## mu[6]        0.17230287 0.16259987 -0.085057655  0.434138295  4350.861 1.0009738
## mu[7]       -0.85826697 0.17099531 -1.126571550 -0.574865565  5371.882 1.0004911
## mu[8]       -0.27400146 0.21633306 -0.619410005  0.074384511  3869.368 1.0012436
## mu[9]       -1.87525316 0.41663095 -2.522689800 -1.198723950  4519.998 1.0006046
## mu[10]      -0.27326890 0.13637037 -0.488130980 -0.052754201  4623.292 1.0003355
## mu[11]       0.05226838 0.12746952 -0.148798045  0.256310440  4691.476 1.0008445
## mu[12]      -0.39201620 0.31717849 -0.894887310  0.118153880  3738.216 1.0012329
## mu[13]       1.43094870 0.28588414  0.971899085  1.877583300  5869.527 0.9998131
## mu[14]      -0.55224705 0.12743981 -0.750853210 -0.342908360  5528.661 1.0004821
## mu[15]       0.11344726 0.10614831 -0.056418403  0.282030470  5831.978 0.9997384
## mu[16]       0.28713622 0.11475253  0.102353375  0.469533145  6013.011 0.9998136
## mu[17]       0.49220996 0.13847435  0.269886905  0.712513055  6001.160 0.9996939
## mu[18]       0.59156544 0.15277294  0.346277680  0.833088225  5950.437 0.9996553
## mu[19]       0.02840800 0.09759959 -0.125811515  0.183895685  5927.779 1.0001531
## mu[20]      -0.32894313 0.26529936 -0.752990010  0.096103720  3909.196 1.0007467
## mu[21]      -0.69253148 0.15334118 -0.932154705 -0.438970650  5108.811 1.0006518
## mu[22]      -1.31888760 0.24771995 -1.703738700 -0.911026375  5278.152 1.0004128
## mu[23]      -0.28122236 0.15217999 -0.522699040 -0.034297223  4413.093 1.0004192
## mu[24]      -0.25142828 0.20124886 -0.572787130  0.072783579  4053.505 1.0005987
## mu[25]       0.05661140 0.10826571 -0.116867440  0.228773165  5441.327 0.9998241
## mu[26]       0.14324135 0.14067471 -0.081770550  0.365280320  4623.482 1.0000246
## mu[27]       0.09276826 0.13500908 -0.122884310  0.308316005  4653.149 1.0000333
## mu[28]       0.25691307 0.13267203  0.044161392  0.465683770  5113.609 0.9997937
## mu[29]      -0.47357056 0.13634268 -0.688323375 -0.251377105  5031.458 1.0003721
## mu[30]      -0.94373530 0.19389867 -1.248703450 -0.630089665  5283.769 1.0003941
## mu[31]       0.07410902 0.09786699 -0.082064206  0.230266905  6165.449 0.9999040
## mu[32]      -1.25409823 0.24696205 -1.637439500 -0.849292205  5031.086 1.0005098
## mu[33]       0.12299141 0.10104808 -0.038110109  0.283948085  6173.788 0.9997846
## mu[34]       0.41873461 0.26078475  0.001529521  0.837020585  4065.971 1.0010648
## mu[35]      -0.22597720 0.14407413 -0.455028630  0.007344778  4460.882 1.0003586
## mu[36]       0.81254612 0.18552041  0.514135305  1.102662650  5909.486 0.9997949
## mu[37]      -0.04751616 0.10524679 -0.214064505  0.120238970  5364.296 0.9999420
## mu[38]      -0.63930606 0.16607356 -0.899365420 -0.371697955  4882.600 1.0004483
## mu[39]      -1.18496593 0.22064639 -1.528772200 -0.819949965  5501.661 1.0003547
## mu[40]      -0.25577129 0.10903499 -0.426521335 -0.078303428  5324.682 1.0001939
## mu[41]       0.16710174 0.10842556 -0.005111127  0.339597535  5981.592 0.9996926
## mu[42]       0.35149647 0.15680375  0.099315038  0.599341595  4768.295 0.9999118
## mu[43]       0.38490101 0.12533622  0.183672870  0.583733750  6039.993 0.9996647
## mu[44]       1.44251261 0.33227378  0.904220555  1.962507300  5200.977 1.0003215
## mu[45]      -0.52722504 0.14344357 -0.750844870 -0.295172065  5018.435 1.0003943
## mu[46]      -0.21759467 0.11461005 -0.395860855 -0.034042869  4866.196 1.0008638
## mu[47]       0.04113354 0.10978300 -0.132190430  0.216880695  5204.079 1.0005660
```

```
## mu[48]      0.49380065 0.13872150  0.270663985  0.714217785  5993.524 0.9997156
## mu[49]     -0.22120512 0.13465949 -0.435008260 -0.004322070  4595.733 1.0003027
## mu[50]      1.02006869 0.36874635  0.429420410  1.606307000  4285.306 1.0008725
```

```
dashboard(m0)
```




```
post <- extract.samples(m0)
df$D_true <- post$D_true %>% apply(2,mean)
df$D_obs  <- d$D_obs
df$A      <- d$A

ggplot(df) +
    geom_point(aes(x=A, y=D_obs , colour='Obs')) +
    geom_point(aes(x=A, y=D_true, colour='Pred')) +
    geom_linerange(aes(x=A, ymin=D_obs, ymax=D_true)) +
    scale_colour_tableau()
```



We see shrinkage towards the mean, just as we see with partial pooling. However the overall strength of the regression shouldn't have changed that much, the shrinkage has followed the rules. So lets go further and add M error to our model.

$$
\begin{aligned}
D_i^{true} &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_M M_i + \beta_A A_i \\
\\
D_i^{obs} &\sim \text{Normal}(D_i^{true}, S_i) \\
\\
M_i^{true} &\sim \text{Normal}(\nu_i, \tau) \\
\nu_i &= \alpha_M + \beta_{A,M} A_i \\
\\
M^{obs} &\sim \text{Normal}(D_i^{true}, T_i)
\end{aligned}
$$

Then it is a simple as writing up the dual models. Yes this is exactly a partial pooling.

```
d <- list(
    D_obs = standardize( df$Divorce )
,   D_std = df$Divorce.SE / sd(df$Divorce)
,   M_obs = standardize( df$Marriage )
,   M_std = df$Marriage.SE / sd(df$Marriage)
,   A     = standardize( df$MedianAgeMarriage )
```

```r
,   N     = nrow(df)
)

m1 <- cstan(file='../models/l17_m1.stan', data=d, chains=4, cores=4, threads=2, iter=4000)
```

```
## Running MCMC with 4 parallel chains, with 2 thread(s) per chain...
##
## Chain 1 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 1 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 1 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 1 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 1 Iteration:  400 / 4000 [ 10%]  (Warmup)

## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 1 Exception: exponential_lpdf: Random variable is -683.542, but must be nonnegativ

## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 1 but if this warning occurs often then your model may be either severely ill-cond

## Chain 1

## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 1 Exception: exponential_lpdf: Random variable is -4.0303, but must be nonnegative

## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 1 but if this warning occurs often then your model may be either severely ill-cond

## Chain 1

## Chain 2 Rejecting initial value:

## Chain 2   Error evaluating the log probability at the initial value.

## Chain 2 Exception: exponential_lpdf: Random variable is -1.91853, but must be nonnegativ
## Chain 2 Exception: exponential_lpdf: Random variable is -1.91853, but must be nonnegativ

## Chain 2 Rejecting initial value:

## Chain 2   Error evaluating the log probability at the initial value.

## Chain 2 Exception: exponential_lpdf: Random variable is -0.329903, but must be nonnegati
## Chain 2 Exception: exponential_lpdf: Random variable is -0.329903, but must be nonnegati

## Chain 2 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 2 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 2 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 2 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 2 Iteration:  400 / 4000 [ 10%]  (Warmup)

## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected b
```

```
## Chain 2 Exception: exponential_lpdf: Random variable is -1.44075, but must be nonnegativ
## Chain 2 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 2 but if this warning occurs often then your model may be either severely ill-conc
## Chain 2
## Chain 3 Rejecting initial value:
## Chain 3   Error evaluating the log probability at the initial value.
## Chain 3 Exception: exponential_lpdf: Random variable is -0.547491, but must be nonnegati
## Chain 3 Exception: exponential_lpdf: Random variable is -0.547491, but must be nonnegati
## Chain 3 Rejecting initial value:
## Chain 3   Error evaluating the log probability at the initial value.
## Chain 3 Exception: exponential_lpdf: Random variable is -0.635733, but must be nonnegati
## Chain 3 Exception: exponential_lpdf: Random variable is -0.635733, but must be nonnegati
## Chain 3 Rejecting initial value:
## Chain 3   Error evaluating the log probability at the initial value.
## Chain 3 Exception: exponential_lpdf: Random variable is -1.22837, but must be nonnegativ
## Chain 3 Exception: exponential_lpdf: Random variable is -1.22837, but must be nonnegativ
## Chain 3 Rejecting initial value:
## Chain 3   Error evaluating the log probability at the initial value.
## Chain 3 Exception: exponential_lpdf: Random variable is -0.323202, but must be nonnegati
## Chain 3 Exception: exponential_lpdf: Random variable is -0.323202, but must be nonnegati
## Chain 3 Rejecting initial value:
## Chain 3   Error evaluating the log probability at the initial value.
## Chain 3 Exception: exponential_lpdf: Random variable is -1.9036, but must be nonnegative
## Chain 3 Exception: exponential_lpdf: Random variable is -1.9036, but must be nonnegative
## Chain 3 Rejecting initial value:
## Chain 3   Error evaluating the log probability at the initial value.
## Chain 3 Exception: exponential_lpdf: Random variable is -0.438092, but must be nonnegati
## Chain 3 Exception: exponential_lpdf: Random variable is -0.438092, but must be nonnegati
## Chain 3 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 3 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 3 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 3 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 3 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected b
```

```
## Chain 3 Exception: exponential_lpdf: Random variable is -1.56858, but must be nonnegativ
## Chain 3 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 3 but if this warning occurs often then your model may be either severely ill-conc
## Chain 3
## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 3 Exception: exponential_lpdf: Random variable is -2.49219, but must be nonnegativ
## Chain 3 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 3 but if this warning occurs often then your model may be either severely ill-conc
## Chain 3
## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 3 Exception: exponential_lpdf: Random variable is -12.9336, but must be nonnegativ
## Chain 3 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 3 but if this warning occurs often then your model may be either severely ill-conc
## Chain 3
## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 3 Exception: exponential_lpdf: Random variable is -0.0228516, but must be nonnegat
## Chain 3 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 3 but if this warning occurs often then your model may be either severely ill-conc
## Chain 3
## Chain 4 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 4 Iteration:  100 / 4000 [  2%]  (Warmup)
## Chain 4 Iteration:  200 / 4000 [  5%]  (Warmup)
## Chain 4 Iteration:  300 / 4000 [  7%]  (Warmup)
## Chain 4 Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 4 Exception: exponential_lpdf: Random variable is -60.4626, but must be nonnegativ
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 4 but if this warning occurs often then your model may be either severely ill-conc
## Chain 4
## Chain 1 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 1 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 1 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 1 Iteration:  800 / 4000 [ 20%]  (Warmup)
```

```
## Chain 1 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 1 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 1 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 2 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 2 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 2 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 2 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 2 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 2 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 2 Iteration: 1100 / 4000 [ 27%]  (Warmup)

## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 2 Exception: exponential_lpdf: Random variable is -0.139128, but must be nonnegati

## Chain 2 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 2 but if this warning occurs often then your model may be either severely ill-cond

## Chain 2

## Chain 3 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 3 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 3 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 3 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 3 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 3 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 4 Iteration:  500 / 4000 [ 12%]  (Warmup)
## Chain 4 Iteration:  600 / 4000 [ 15%]  (Warmup)
## Chain 4 Iteration:  700 / 4000 [ 17%]  (Warmup)
## Chain 4 Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 4 Iteration:  900 / 4000 [ 22%]  (Warmup)
## Chain 4 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 1 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 1 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 1 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 1 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 1 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 2 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 2 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 2 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 2 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 3 Iteration: 1100 / 4000 [ 27%]  (Warmup)
## Chain 3 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 3 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 3 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 3 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 3 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 4 Iteration: 1100 / 4000 [ 27%]  (Warmup)
```

```
## Chain 4 Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 4 Iteration: 1300 / 4000 [ 32%]  (Warmup)
## Chain 4 Iteration: 1400 / 4000 [ 35%]  (Warmup)
## Chain 1 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 1 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 1 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 1 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 2 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 2 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 3 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 3 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 3 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 3 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 3 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1500 / 4000 [ 37%]  (Warmup)
## Chain 4 Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 4 Iteration: 1700 / 4000 [ 42%]  (Warmup)
## Chain 1 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 1 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 1 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 1 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 1 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 2 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 2 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 2 Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 2 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 2 Iteration: 2200 / 4000 [ 55%]  (Sampling)

## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 2 Exception: exponential_lpdf: Random variable is -1.00672, but must be nonnegativ

## Chain 2 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 2 but if this warning occurs often then your model may be either severely ill-conc

## Chain 2

## Chain 3 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 3 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 3 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 3 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 3 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 4 Iteration: 1800 / 4000 [ 45%]  (Warmup)
## Chain 4 Iteration: 1900 / 4000 [ 47%]  (Warmup)
## Chain 4 Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 4 Iteration: 2001 / 4000 [ 50%]  (Sampling)
```

```
## Chain 4 Iteration: 2100 / 4000 [ 52%]  (Sampling)
## Chain 4 Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 1 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 1 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 1 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 1 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 1 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 2 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 2 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 2 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 2 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 3 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 3 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 3 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 3 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 3 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 4 Iteration: 2300 / 4000 [ 57%]  (Sampling)
## Chain 4 Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 4 Iteration: 2500 / 4000 [ 62%]  (Sampling)
## Chain 4 Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 1 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 1 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 1 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 1 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 1 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 2 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 2 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 2 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 2 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 2 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 3 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 3 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 3 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 3 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 3 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 3 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 4 Iteration: 2700 / 4000 [ 67%]  (Sampling)
## Chain 4 Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 4 Iteration: 2900 / 4000 [ 72%]  (Sampling)
## Chain 4 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 4 Iteration: 3100 / 4000 [ 77%]  (Sampling)
## Chain 1 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 1 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 1 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 1 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 1 Iteration: 4000 / 4000 [100%]  (Sampling)
```

```
## Chain 2 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 2 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 2 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 2 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 2 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 3 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 3 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 3 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 3 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 4 Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 4 Iteration: 3300 / 4000 [ 82%]  (Sampling)
## Chain 4 Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 4 Iteration: 3500 / 4000 [ 87%]  (Sampling)
## Chain 4 Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 1 finished in 0.8 seconds.
## Chain 3 finished in 0.8 seconds.
## Chain 2 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 2 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 2 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 2 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 4 Iteration: 3700 / 4000 [ 92%]  (Sampling)
## Chain 4 Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 4 Iteration: 3900 / 4000 [ 97%]  (Sampling)
## Chain 4 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 2 finished in 0.9 seconds.
## Chain 4 finished in 0.9 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.9 seconds.
## Total execution time: 1.0 seconds.
```

```
precis(m1, depth=2)
```

```
##                    mean         sd          5.5%       94.5%     n_eff
## D_true[1]   1.134621362 0.37426718  0.5517359500  1.749407500 10273.730
## D_true[2]   0.700111733 0.52363517 -0.1177343950  1.547246750 13241.917
## D_true[3]   0.423242154 0.33364763 -0.1102239150  0.967498925 13046.074
## D_true[4]   1.454058809 0.46621338  0.7141527950  2.206179900 12481.066
## D_true[5]  -0.896225044 0.12892194 -1.1041922000 -0.690617305 15388.834
## D_true[6]   0.691368072 0.39251391  0.0736190435  1.316402100 12410.934
## D_true[7]  -1.353742196 0.33955478 -1.8991816000 -0.818933625 13604.912
## D_true[8]  -0.326714578 0.46412501 -1.0562545500  0.416486360 14162.210
## D_true[9]  -1.879228948 0.56208185 -2.7803177000 -0.985054450 11084.845
## D_true[10] -0.626330472 0.16679508 -0.8906578700 -0.358350710 16057.073
## D_true[11]  0.779638402 0.28044426  0.3332335050  1.235927400 12186.852
## D_true[12] -0.519337738 0.45122973 -1.2389689500  0.196820525 13652.181
```

```
## D_true[13]  0.229731637 0.49527256 -0.5875254450  1.001814400  6262.299
## D_true[14] -0.863150256 0.22435846 -1.2268805500 -0.501293535 16978.002
## D_true[15]  0.540613839 0.30140375  0.0659366495  1.034043550 12760.739
## D_true[16]  0.296800877 0.37903025 -0.2974132550  0.896402270 16363.849
## D_true[17]  0.510960665 0.42157337 -0.1686192300  1.182655400 14395.513
## D_true[18]  1.242121784 0.34682804  0.6985832050  1.811154300 12324.191
## D_true[19]  0.435542113 0.37360277 -0.1510135950  1.047961000 12308.578
## D_true[20]  0.258551497 0.53618377 -0.5480274850  1.135893000  7389.754
## D_true[21] -0.548569076 0.31427252 -1.0494165000 -0.039906433 12777.116
## D_true[22] -1.100441461 0.25974257 -1.5172551000 -0.694373375 13572.728
## D_true[23] -0.305020376 0.25533720 -0.7096548400  0.104499620 13848.596
## D_true[24] -1.032751441 0.29383012 -1.5051099000 -0.565507300 12309.240
## D_true[25]  0.412801062 0.40464299 -0.2241039000  1.060793000 13326.928
## D_true[26] -0.053174795 0.31094729 -0.5469783800  0.442607545 15162.158
## D_true[27] -0.024865427 0.48729668 -0.8027943000  0.746764745 14913.230
## D_true[28] -0.156971332 0.39166055 -0.7953711600  0.457552705 13577.497
## D_true[29] -0.299432369 0.48375627 -1.0578311500  0.483687350 14758.971
## D_true[30] -1.802845289 0.23617813 -2.1806392500 -1.433843050 13376.386
## D_true[31]  0.172324383 0.42010702 -0.4834890100  0.853774655 14077.850
## D_true[32] -1.650256868 0.16397769 -1.9093722000 -1.384517800 13811.387
## D_true[33]  0.117054397 0.23998385 -0.2685540100  0.498212685 16539.991
## D_true[34] -0.022107581 0.47731946 -0.7994115250  0.722755830 10463.756
## D_true[35] -0.147990914 0.22833001 -0.5084176500  0.217071220 13069.315
## D_true[36]  1.288987849 0.40750900  0.6376269600  1.951475850 13551.204
## D_true[37]  0.209832350 0.35367226 -0.3497882450  0.789574395 14099.872
## D_true[38] -1.040881072 0.21965135 -1.3902382000 -0.692034940 13683.247
## D_true[39] -0.941815329 0.53317662 -1.7624533000 -0.059006258 11377.706
## D_true[40] -0.679198318 0.31878967 -1.1893215500 -0.179620910 13717.430
## D_true[41]  0.238527296 0.54475356 -0.6269338050  1.116153150 13693.159
## D_true[42]  0.701157089 0.34180987  0.1636679000  1.259904900 13694.434
## D_true[43]  0.198381412 0.18009776 -0.0917631260  0.484645545 17435.993
## D_true[44]  0.873183746 0.42591950  0.1725372250  1.545471550  8795.854
## D_true[45] -0.432910091 0.52481367 -1.2476244000  0.421928405 13065.465
## D_true[46] -0.368985086 0.25647454 -0.7807645300  0.039650097 12677.994
## D_true[47]  0.149993754 0.30127298 -0.3255424250  0.637963340 16243.794
## D_true[48]  0.569759606 0.45155956 -0.1546294950  1.281405650 13323.276
## D_true[49] -0.650917081 0.27588441 -1.0908610000 -0.218832050 13642.555
## D_true[50]  0.855395735 0.53329648  0.0062570736  1.693670550 12019.630
## M_true[1]   0.180691531 0.26519470 -0.2459218400  0.598608955 13177.449
## M_true[2]   0.655419681 0.39438842  0.0410037705  1.297476050 13392.433
## M_true[3]   0.057832602 0.21777230 -0.2932669250  0.404632575 14546.227
## M_true[4]   1.265819897 0.32678981  0.7646004800  1.797780350 12045.135
## M_true[5]  -0.287029458 0.09892690 -0.4458588600 -0.129179215 13865.634
## M_true[6]   0.607939469 0.26658648  0.1810908700  1.039088150 11942.373
## M_true[7]  -0.864797069 0.23116856 -1.2270114500 -0.488595270 13556.663
## M_true[8]  -0.116751731 0.38730613 -0.7274983450  0.501779190 13891.059
```

```
## M_true[9]  -1.629503354 0.41294260 -2.2760069500 -0.953832160 10341.845
## M_true[10] -0.761988880 0.14540090 -0.9931045550 -0.534982615 15685.138
## M_true[11]  0.432879521 0.19363506  0.1217102250  0.749344665 11552.824
## M_true[12] -0.046034375 0.38510711 -0.6433779650  0.590562825 10400.527
## M_true[13]  1.325895772 0.36105887  0.7538799550  1.904432000  9063.793
## M_true[14] -0.591669725 0.14652976 -0.8248557450 -0.353395815 15336.664
## M_true[15] -0.035280529 0.18913949 -0.3376905200  0.264124345 14723.424
## M_true[16]  0.303882502 0.28655630 -0.1505130650  0.758542440 12573.666
## M_true[17]  0.489543086 0.28967440  0.0321775475  0.948389290 14922.270
## M_true[18]  0.586851076 0.23986240  0.2062790600  0.965047750 13808.202
## M_true[19]  0.092158888 0.25725453 -0.3148539600  0.505620520 17014.657
## M_true[20] -1.071977247 0.29695433 -1.5492135500 -0.605942865 10121.828
## M_true[21] -0.557001546 0.22649609 -0.9139493750 -0.195580960 14329.451
## M_true[22] -1.175514928 0.16776364 -1.4382926500 -0.908710600 14281.206
## M_true[23] -0.845216286 0.17049855 -1.1230771500 -0.572881570 13657.768
## M_true[24] -1.093826107 0.19177498 -1.3955441000 -0.787193485 13347.454
## M_true[25] -0.073260188 0.29495074 -0.5401893300  0.398640905 13820.046
## M_true[26] -0.296098814 0.19358049 -0.6033755150  0.013794007 15817.210
## M_true[27] -0.100859637 0.35298914 -0.6584499500  0.462256705 11995.833
## M_true[28] -0.005869327 0.28595969 -0.4626281100  0.444180980 14974.362
## M_true[29] -0.676272341 0.32548749 -1.2021332000 -0.161167025 15135.732
## M_true[30] -1.366059466 0.14948400 -1.6040655000 -1.128728550 13735.583
## M_true[31]  0.054365168 0.31928339 -0.4612361350  0.557725365 13646.028
## M_true[32] -0.919325569 0.11970415 -1.1060705500 -0.726449900 14783.777
## M_true[33]  0.074207637 0.21974021 -0.2736199400  0.429352625 15879.412
## M_true[34]  0.576068760 0.40071508 -0.0512272120  1.228914100 12845.255
## M_true[35] -0.763908840 0.14928291 -1.0047787000 -0.525460560 12583.289
## M_true[36]  0.920354791 0.26951005  0.4879676750  1.352536600 15442.196
## M_true[37] -0.225957780 0.23686767 -0.6016534200  0.156181605 14181.869
## M_true[38] -1.173742779 0.12274050 -1.3732805500 -0.978758285 14905.317
## M_true[39] -1.260480039 0.35535209 -1.8233793500 -0.686391005 12559.763
## M_true[40] -0.472903022 0.24800688 -0.8705942100 -0.069909799 15356.764
## M_true[41]  0.099837428 0.37274120 -0.4896214000  0.685406610 13260.137
## M_true[42] -0.050122966 0.19882448 -0.3735434000  0.268458425 14281.655
## M_true[43]  0.356585664 0.15065192  0.1167839150  0.599257005 16182.161
## M_true[44]  1.816866942 0.34801034  1.2732260000  2.377603650  8798.998
## M_true[45] -0.682363295 0.36195075 -1.2672381500 -0.106826910 14611.407
## M_true[46]  0.008120311 0.19258348 -0.3011057600  0.320116125 13784.765
## M_true[47]  0.233822146 0.22790016 -0.1299739700  0.597511250 17625.256
## M_true[48]  0.498159275 0.30737263  0.0082633134  0.991080980 12559.165
## M_true[49] -0.679417225 0.18743839 -0.9788828300 -0.379102220 14008.023
## M_true[50]  1.143659965 0.42751606  0.4730490600  1.835752750 11371.808
## alpha_D    -0.024732876 0.09700397 -0.1794993150  0.129659880  8107.854
## alpha_M    -0.111039171 0.07494790 -0.2279881300  0.008360996 10074.798
## betaD_A    -0.468928745 0.19490452 -0.7781656900 -0.158771230  5773.225
## betaD_M     0.297639913 0.25770613 -0.1165489600  0.710031245  4372.645
```

```
## betaM_A    -0.663703319 0.08467812 -0.7989320700 -0.528922175  8303.352
## sigma       0.557352681 0.11003110  0.3921989200  0.739009145  2985.633
## tau         0.437833216 0.07049379  0.3338967900  0.556557900  3495.701
## mu[1]       0.323848589 0.15192145  0.0805904655  0.562638225  8796.016
## mu[2]       0.490643327 0.20987063  0.1895015950  0.852616610  7883.438
## mu[3]       0.090547983 0.12707424 -0.1039189100  0.302455475  8757.145
## mu[4]       1.015890368 0.25749108  0.6171440600  1.444280550  6375.812
## mu[5]      -0.393192842 0.12785699 -0.5905861350 -0.187485185  7981.133
## mu[6]       0.290976223 0.19904112  0.0065860678  0.633817455  6570.365
## mu[7]      -0.870826969 0.18492015 -1.1585420000 -0.572010855  8107.548
## mu[8]      -0.269915495 0.18769960 -0.5356822750  0.046124995  8645.022
## mu[9]      -1.884912506 0.37703046 -2.4706084000 -1.270551750  8231.118
## mu[10]     -0.381963527 0.16498054 -0.6473867400 -0.123510225  6043.458
## mu[11]      0.164839758 0.17171808 -0.0878878065  0.455279450  5731.840
## mu[12]     -0.367172898 0.21629245 -0.6761984750  0.004491119  7188.909
## mu[13]      1.416153638 0.30330658  0.9214648750  1.896976650  9254.360
## mu[14]     -0.559144013 0.13524028 -0.7713104500 -0.340639070  8791.849
## mu[15]      0.101715669 0.12475805 -0.0987785195  0.294508445  9361.540
## mu[16]      0.310983320 0.15494835  0.0730471145  0.564515880  9031.791
## mu[17]      0.516247757 0.17478535  0.2418150850  0.797667475  8924.256
## mu[18]      0.628936719 0.17464761  0.3527931500  0.907604750  8460.272
## mu[19]      0.064679166 0.14158680 -0.1435885050  0.302494390  9075.921
## mu[20]     -0.458281146 0.23324253 -0.8377755400 -0.106501780  5742.571
## mu[21]     -0.660441064 0.17282235 -0.9237619550 -0.380404420  8159.574
## mu[22]     -1.295410067 0.24925105 -1.6815849500 -0.886537475  8388.517
## mu[23]     -0.403031648 0.18054099 -0.6939996550 -0.119569660  5805.599
## mu[24]     -0.443482799 0.24806031 -0.8525864900 -0.055508691  5017.005
## mu[25]      0.056427622 0.14587440 -0.1728853500  0.289705690  9827.512
## mu[26]      0.058173099 0.16408182 -0.2150299100  0.303288090  6974.855
## mu[27]      0.077038651 0.16972165 -0.2044077100  0.324119695  9113.622
## mu[28]      0.215126722 0.16468613 -0.0631731275  0.459185960  8335.901
## mu[29]     -0.502186415 0.17332564 -0.7813119750 -0.240478690  8565.068
## mu[30]     -1.055198304 0.21647400 -1.4010887000 -0.705966610  6444.493
## mu[31]      0.088102060 0.15569593 -0.1485730800  0.332976255  9037.297
## mu[32]     -1.184781591 0.25059682 -1.5787966000 -0.775979080  8036.190
## mu[33]      0.129111281 0.12766886 -0.0722115160  0.331073765  8606.156
## mu[34]      0.413612393 0.19586282  0.1239662550  0.735313100  7880.067
## mu[35]     -0.342213261 0.17157218 -0.6213227750 -0.071570932  5734.311
## mu[36]      0.877947923 0.21288437  0.5447178350  1.224660450  8372.148
## mu[37]     -0.066889425 0.13250121 -0.2779994250  0.135319570  9675.725
## mu[38]     -0.768938273 0.19682758 -1.0882326500 -0.452070765  5788.815
## mu[39]     -1.199496139 0.25071875 -1.5879440000 -0.798299735  8642.972
## mu[40]     -0.298542777 0.14467860 -0.5393328150 -0.083578907  6977.630
## mu[41]      0.176088632 0.17517115 -0.0980795820  0.449427805  9014.840
## mu[42]      0.286801188 0.16681942  0.0083275232  0.542245895  7800.550
## mu[43]      0.401479181 0.13596698  0.1855816450  0.621117305  9546.602
```

```
## mu[44]       1.529752187 0.31236575  1.0296529500  2.022700400  8282.440
## mu[45]      -0.542834001 0.18211193 -0.8386909950 -0.266670860  8190.995
## mu[46]      -0.156204228 0.13889533 -0.3639325500  0.080528507  7298.494
## mu[47]       0.101881383 0.14339826 -0.1060580150  0.343252440  8182.835
## mu[48]       0.519333830 0.17795379  0.2389865050  0.811781475  9045.162
## mu[49]      -0.319841506 0.16374036 -0.5915484600 -0.072860498  5953.574
## mu[50]       1.002343822 0.26119071  0.6054456700  1.427259350  7690.597
## nu[1]        0.291357187 0.09494231  0.1409983250  0.444133070  9699.821
## nu[2]        0.344725405 0.09929996  0.1870573900  0.505173235  9635.786
## nu[3]        0.024516098 0.07856648 -0.0988545305  0.150502815  9831.689
## nu[4]        0.825039368 0.14714280  0.5943746550  1.064329900  9372.561
## nu[5]       -0.509166077 0.08628009 -0.6442398700 -0.369744835  9719.172
## nu[6]        0.077884323 0.08096614 -0.0493038210  0.207562995  9775.345
## nu[7]       -0.936111852 0.12303499 -1.1292221000 -0.743526415  9336.485
## nu[8]       -0.402429643 0.08030039 -0.5296149050 -0.273910975  9855.449
## nu[9]       -2.056844376 0.25212480 -2.4579915500 -1.654772400  8769.059
## nu[10]      -0.295693199 0.07630824 -0.4162759850 -0.173784395  9979.592
## nu[11]      -0.028852116 0.07669806 -0.1491472750  0.093821183 10049.761
## nu[12]      -0.562534291 0.08989655 -0.7038905850 -0.418663305  9654.097
## nu[13]       1.412089692 0.21508420  1.0762080000  1.760843850  9052.113
## nu[14]      -0.615902511 0.09386825 -0.7635999050 -0.465919130  9593.373
## nu[15]       0.077884323 0.08096614 -0.0493038210  0.207562995  9775.345
## nu[16]       0.237988974 0.09088634  0.0944463975  0.384400050  9774.712
## nu[17]       0.451461841 0.10877286  0.2796306850  0.626676305  9535.969
## nu[18]       0.558198275 0.11905243  0.3701502850  0.749641140  9466.054
## nu[19]      -0.028852116 0.07669806 -0.1491472750  0.093821183 10049.761
## nu[20]      -0.295693199 0.07630824 -0.4162759850 -0.173784395  9979.592
## nu[21]      -0.776007158 0.10750425 -0.9446821650 -0.605887140  9442.754
## nu[22]      -1.416425789 0.17574396 -1.6931349500 -1.138573950  9177.494
## nu[23]      -0.295693199 0.07630824 -0.4162759850 -0.173784395  9979.592
## nu[24]      -0.242324988 0.07516269 -0.3601180150 -0.122381095 10027.025
## nu[25]       0.024516098 0.07856648 -0.0988545305  0.150502815  9831.689
## nu[26]       0.131252532 0.08385142 -0.0005996638  0.266809105  9829.541
## nu[27]       0.077884323 0.08096614 -0.0493038210  0.207562995  9775.345
## nu[28]       0.237988974 0.09088634  0.0944463975  0.384400050  9774.712
## nu[29]      -0.509166077 0.08628009 -0.6442398700 -0.369744835  9719.172
## nu[30]      -0.989479993 0.12851658 -1.1913960500 -0.787929720  9309.048
## nu[31]       0.024516098 0.07856648 -0.0988545305  0.150502815  9831.689
## nu[32]      -1.363057545 0.16960319 -1.6300598000 -1.094837250  9187.999
## nu[33]       0.077884323 0.08096614 -0.0493038210  0.207562995  9775.345
## nu[34]       0.291357187 0.09494231  0.1409983250  0.444133070  9699.821
## nu[35]      -0.242324988 0.07516269 -0.3601180150 -0.122381095 10027.025
## nu[36]       0.771671148 0.14131544  0.5491987500  1.000937700  9384.806
## nu[37]      -0.082220332 0.07540037 -0.1999146050  0.038194361 10070.586
## nu[38]      -0.669270730 0.09815208 -0.8236412850 -0.513324980  9537.789
## nu[39]      -1.256321128 0.15748646 -1.5038731000 -1.009259450  9212.964
```

```
## nu[40]      -0.295693199 0.07630824 -0.4162759850 -0.173784395  9979.592
## nu[41]       0.131252532 0.08385142 -0.0005996638  0.266809105  9829.541
## nu[42]       0.344725405 0.09929996  0.1870573900  0.505173235  9635.786
## nu[43]       0.344725405 0.09929996  0.1870573900  0.505173235  9635.786
## nu[44]       1.358721506 0.20871102  1.0331689000  1.696832750  9075.019
## nu[45]      -0.562534291 0.08989655 -0.7038905850 -0.418663305  9654.097
## nu[46]      -0.295693199 0.07630824 -0.4162759850 -0.173784395  9979.592
## nu[47]      -0.028852116 0.07669806 -0.1491472750  0.093821183 10049.761
## nu[48]       0.451461841 0.10877286  0.2796306850  0.626676305  9535.969
## nu[49]      -0.242324988 0.07516269 -0.3601180150 -0.122381095 10027.025
## nu[50]       0.878407563 0.15305122  0.6380744500  1.127010550  9355.494
##                 Rhat4
## D_true[1]  0.9997342
## D_true[2]  0.9996427
## D_true[3]  0.9995994
## D_true[4]  0.9999813
## D_true[5]  1.0001843
## D_true[6]  0.9996689
## D_true[7]  0.9998109
## D_true[8]  0.9997808
## D_true[9]  0.9999167
## D_true[10] 0.9997423
## D_true[11] 0.9997411
## D_true[12] 0.9999113
## D_true[13] 1.0000918
## D_true[14] 0.9997086
## D_true[15] 0.9998978
## D_true[16] 0.9997096
## D_true[17] 0.9999113
## D_true[18] 0.9999021
## D_true[19] 0.9995690
## D_true[20] 1.0001361
## D_true[21] 0.9996223
## D_true[22] 1.0000213
## D_true[23] 0.9998189
## D_true[24] 0.9995794
## D_true[25] 0.9999012
## D_true[26] 1.0000885
## D_true[27] 0.9997554
## D_true[28] 0.9997754
## D_true[29] 0.9999444
## D_true[30] 0.9996723
## D_true[31] 0.9996451
## D_true[32] 0.9996954
## D_true[33] 0.9997358
## D_true[34] 1.0004006
```

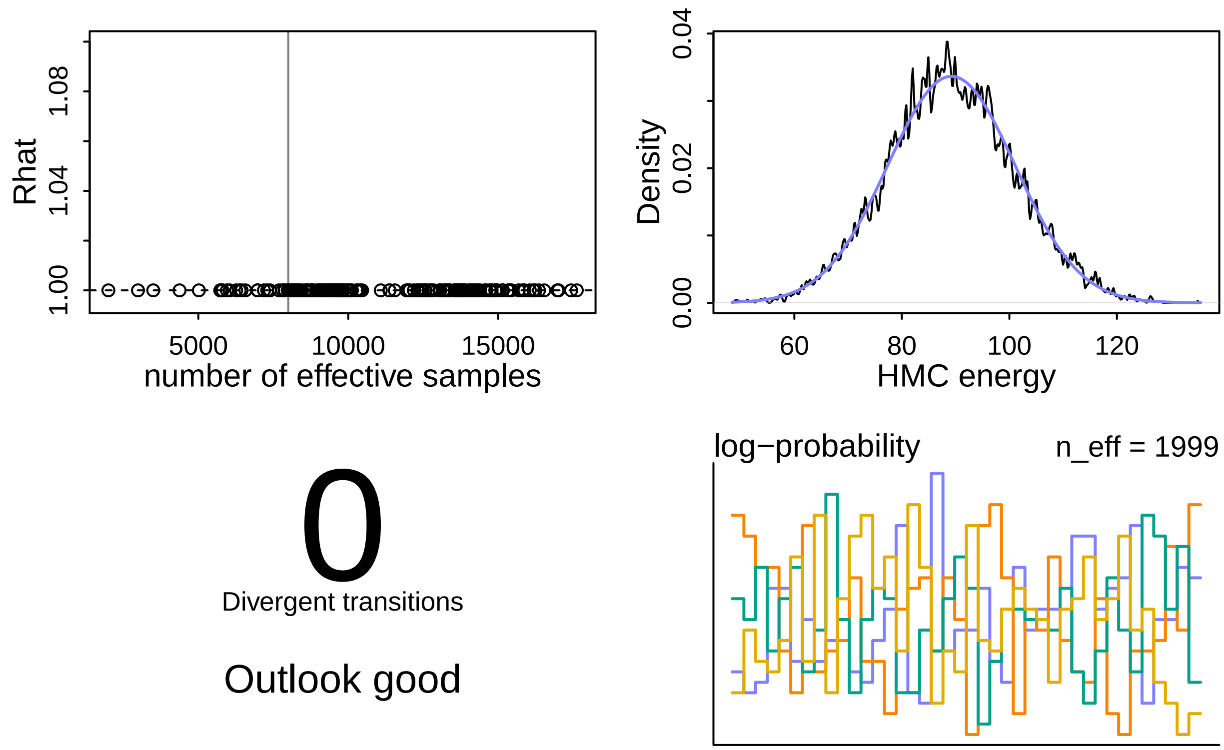
```
## D_true[35] 0.9998836
## D_true[36] 0.9999230
## D_true[37] 0.9997966
## D_true[38] 0.9997939
## D_true[39] 0.9996941
## D_true[40] 0.9995789
## D_true[41] 0.9998110
## D_true[42] 0.9998350
## D_true[43] 1.0000563
## D_true[44] 0.9997707
## D_true[45] 0.9997104
## D_true[46] 1.0000912
## D_true[47] 0.9998992
## D_true[48] 0.9998117
## D_true[49] 0.9996801
## D_true[50] 0.9995814
## M_true[1]  0.9997264
## M_true[2]  0.9997962
## M_true[3]  0.9998866
## M_true[4]  0.9996545
## M_true[5]  0.9996851
## M_true[6]  1.0000208
## M_true[7]  1.0001306
## M_true[8]  0.9996507
## M_true[9]  0.9997639
## M_true[10] 0.9996117
## M_true[11] 0.9997240
## M_true[12] 0.9999809
## M_true[13] 0.9999567
## M_true[14] 0.9998267
## M_true[15] 0.9997792
## M_true[16] 0.9998735
## M_true[17] 0.9996904
## M_true[18] 0.9997395
## M_true[19] 0.9998353
## M_true[20] 1.0000173
## M_true[21] 0.9996418
## M_true[22] 0.9997493
## M_true[23] 0.9996449
## M_true[24] 0.9996843
## M_true[25] 0.9996631
## M_true[26] 0.9996264
## M_true[27] 1.0002946
## M_true[28] 0.9997939
## M_true[29] 0.9998754
## M_true[30] 0.9997598
```

```
## M_true[31] 0.9995884
## M_true[32] 0.9998863
## M_true[33] 1.0001547
## M_true[34] 0.9998349
## M_true[35] 0.9997849
## M_true[36] 0.9997435
## M_true[37] 0.9997673
## M_true[38] 1.0001444
## M_true[39] 0.9997149
## M_true[40] 0.9999664
## M_true[41] 0.9997581
## M_true[42] 0.9999565
## M_true[43] 0.9999228
## M_true[44] 0.9997988
## M_true[45] 0.9997551
## M_true[46] 0.9996914
## M_true[47] 0.9997808
## M_true[48] 0.9999247
## M_true[49] 0.9999197
## M_true[50] 0.9999087
## alpha_D    0.9999848
## alpha_M    0.9998435
## betaD_A    1.0001928
## betaD_M    1.0005870
## betaM_A    0.9996098
## sigma      1.0004979
## tau        1.0004674
## mu[1]      1.0000422
## mu[2]      0.9998865
## mu[3]      1.0001901
## mu[4]      1.0003030
## mu[5]      0.9998955
## mu[6]      1.0003405
## mu[7]      1.0000632
## mu[8]      0.9999142
## mu[9]      0.9998564
## mu[10]     1.0002278
## mu[11]     1.0004195
## mu[12]     0.9999694
## mu[13]     0.9997467
## mu[14]     0.9999195
## mu[15]     0.9998318
## mu[16]     1.0005036
## mu[17]     0.9999628
## mu[18]     0.9999277
## mu[19]     0.9999535
```

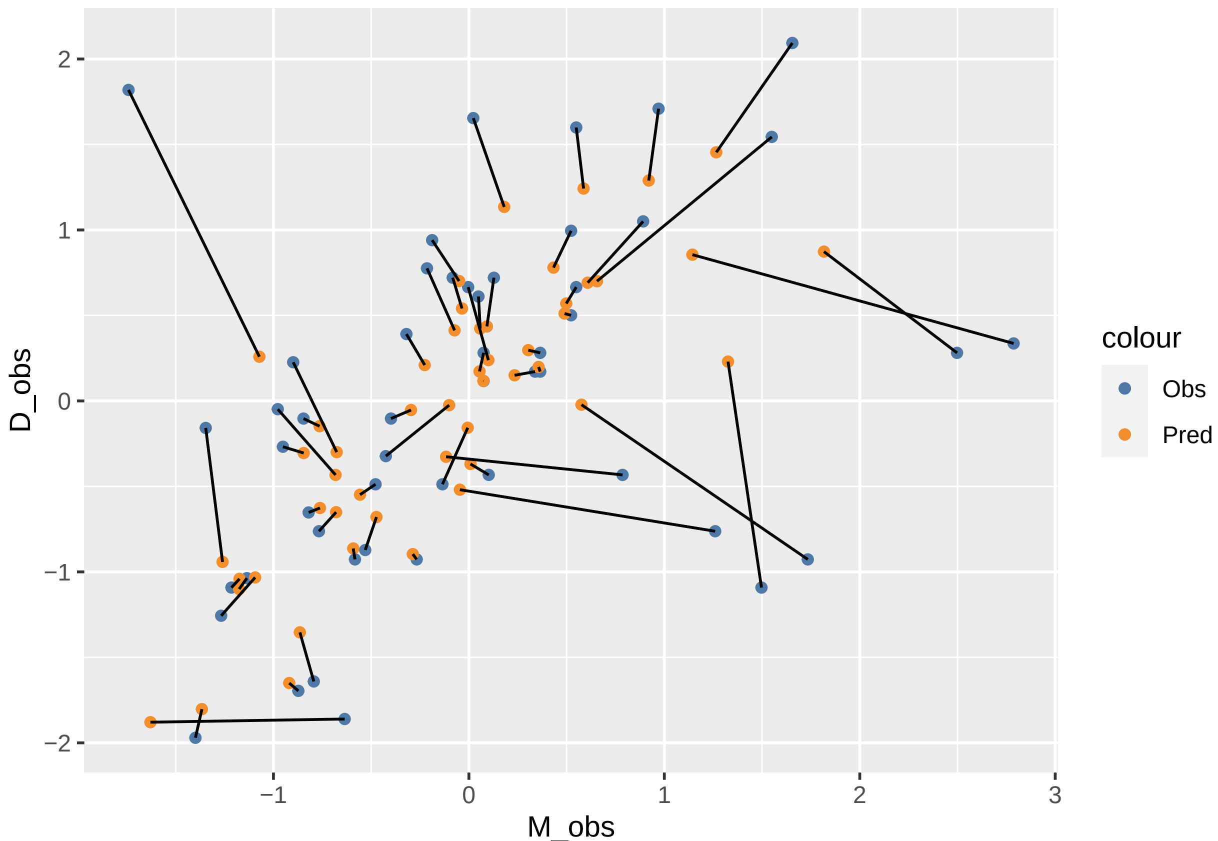
```
## mu[20]      1.0002208
## mu[21]      0.9998142
## mu[22]      0.9997298
## mu[23]      1.0002192
## mu[24]      1.0002094
## mu[25]      0.9997495
## mu[26]      1.0001379
## mu[27]      1.0001130
## mu[28]      0.9998307
## mu[29]      0.9997725
## mu[30]      1.0001878
## mu[31]      0.9997885
## mu[32]      0.9997281
## mu[33]      0.9996290
## mu[34]      0.9997553
## mu[35]      1.0003843
## mu[36]      0.9996854
## mu[37]      0.9998251
## mu[38]      1.0003484
## mu[39]      0.9996420
## mu[40]      1.0001129
## mu[41]      0.9997309
## mu[42]      0.9999693
## mu[43]      1.0002650
## mu[44]      1.0000691
## mu[45]      1.0000093
## mu[46]      0.9997757
## mu[47]      1.0000685
## mu[48]      1.0000124
## mu[49]      1.0002803
## mu[50]      0.9999090
## nu[1]       0.9997707
## nu[2]       0.9997587
## nu[3]       0.9998300
## nu[4]       0.9996852
## nu[5]       0.9997679
## nu[6]       0.9998199
## nu[7]       0.9996779
## nu[8]       0.9997980
## nu[9]       0.9996200
## nu[10]      0.9998245
## nu[11]      0.9998378
## nu[12]      0.9997532
## nu[13]      0.9996495
## nu[14]      0.9997392
## nu[15]      0.9998199
```

```
## nu[16]      0.9997832
## nu[17]      0.9997370
## nu[18]      0.9997186
## nu[19]      0.9998378
## nu[20]      0.9998245
## nu[21]      0.9997038
## nu[22]      0.9996379
## nu[23]      0.9998245
## nu[24]      0.9998343
## nu[25]      0.9998300
## nu[26]      0.9998083
## nu[27]      0.9998199
## nu[28]      0.9997832
## nu[29]      0.9997679
## nu[30]      0.9996711
## nu[31]      0.9998300
## nu[32]      0.9996406
## nu[33]      0.9998199
## nu[34]      0.9997707
## nu[35]      0.9998343
## nu[36]      0.9996907
## nu[37]      0.9998425
## nu[38]      0.9997263
## nu[39]      0.9996470
## nu[40]      0.9998245
## nu[41]      0.9998083
## nu[42]      0.9997587
## nu[43]      0.9997587
## nu[44]      0.9996515
## nu[45]      0.9997532
## nu[46]      0.9998245
## nu[47]      0.9998378
## nu[48]      0.9997370
## nu[49]      0.9998343
## nu[50]      0.9996803
```

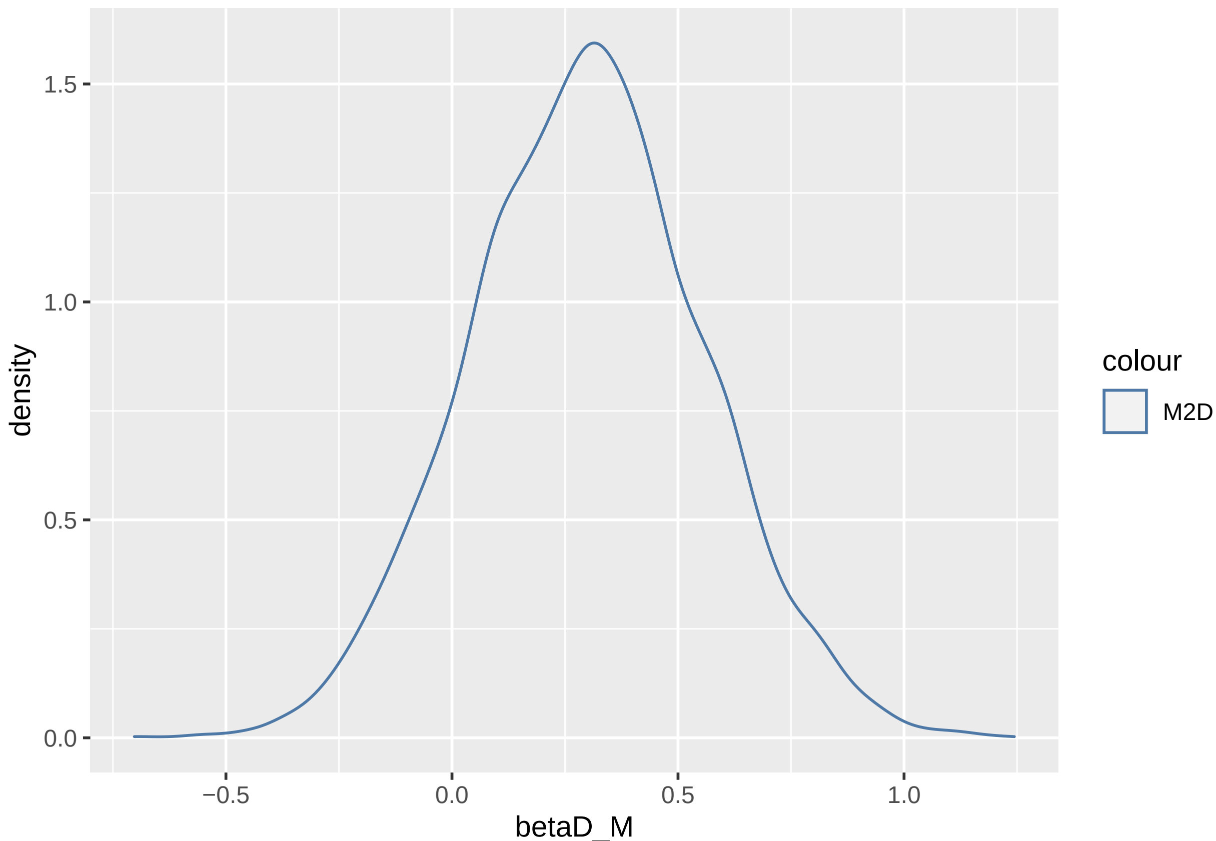
```r
dashboard(m1)
```




```
post <- extract.samples(m1)
df$D_true <- post$D_true %>% apply(2,mean)
df$D_obs  <- d$D_obs
df$M_true <- post$M_true %>% apply(2,mean)
df$M_obs  <- d$M_obs
df$A      <- d$A

ggplot(df) +
    geom_point(aes(x=M_obs, y=D_obs , colour='Obs')) +
    geom_point(aes(x=M_true, y=D_true, colour='Pred')) +
    geom_segment(aes(x=M_obs, xend=M_true, y=D_obs, yend=D_true)) +
    scale_colour_tableau()
```




```
ggplot( data.frame(betaD_M=post$betaD_M) ) +
    geom_density(aes(x=betaD_M, colour='M2D')) +
    scale_colour_tableau()
```



Note that the effect of marriage rate now is positive when compared to our older non error model, the effect of large error states no longer has large has an effect.

Lecture 18: Missing Data

$$
\begin{tikzcd}
 & u \arrow[dl] \arrow[dr] & \\
X \arrow[rr] \arrow[d] & & Y \\
oX & &
\end{tikzcd}
$$

What if we have missing data, ie. some X is missing and we only see the observed oX.

Dog Eating Homework

Basic Dog Eats Homework

$$
\begin{tikzcd}
E \arrow[r] & H \arrow[d] \\
D \arrow[r] & oH
\end{tikzcd}
$$

Suppose we want to see what student effort has on homework quality but dogs eat some of the homework. In this case we can simply drop missing homework as our model says it is done independently. But what about other forms of missing data, oni where dog eating is dependant on some variable.

Dog Eats Homework Dependant on Cause

$$\begin{tikzcd} E \arrow[r] \arrow[d] & H \arrow[d] \\ D \arrow[r] & oH \end{tikzcd}$$

Now for instance the dog eats homework dependant on the effort spent, for instance a neglected dog eats more homework. Now this changes the inference of our model, and we must get the relationship between cause of homework and the dog. If we get it wrong our inference is wrong. However if we model this relationship correctly and/or have additional information this is not a problem.

Dog Eats Homework Dependant on Homework

$$\begin{tikzcd} E \arrow[r] & H \arrow[d] \arrow[dl] \\ D \arrow[r] & oH \end{tikzcd}$$

This is much less benign, however if we can model the dog we can still get effective inference. Otherwise it is pretty much hopeless. For instance we might suspect that bad homework gets fed to the dog. This is however a common class of missing data, for instance survival analysis.

Bayesian Imputation

Now dropping incomplete data can be statistically consistent, it is however not efficient, the non–missing components still tell us something about the data, even if they only inform the population of those values.

One common problem we have is for instance predicting the lifetime of some quantity after some time where some are still remaining (censored observations), that is the endpoint has not been determined for some points but we know it is a least a quantity, by modelling the population of these points we can say something about the eventual population.

Primate Phylogony

Lets us look at some primate data and try to account for missing values that are present in phylogeny often, in the simplest case, all missing data is random, with no bias.

$$
\begin{tikzcd}
mG \arrow[d] & mM \arrow[d] & mB \arrow[d] \\
oG & oM & oB \\
G \arrow[u] \arrow[rr] & & B \arrow[u] \\
& M \arrow[ul] \arrow[ur] \arrow[uu] & \\
& u \arrow[u] \arrow[uul] \arrow[uur] & h \arrow[l]
\end{tikzcd}
$$

We have group size G, Brain size B, and Mass M. We have some missing rates for these values as well as unobserved confounds u determined by some unknown h, as a consequence of our process for acquiring our data. We can use this to make a model if we have some distance matrix d.

$$
\begin{aligned}
B &\sim \mathrm{MVNormal}(\mu_i, K) \\
\mu_i &= \alpha + \beta_G G_i + \beta_M M_i \\
K &= \eta^2 \exp(-\rho d_{ij}) \\
\alpha &\sim \mathrm{Normal}(0, 1) \\
\beta_j &\sim \mathrm{Normal}(0, 0.5) \\
\eta^2 &\sim \mathrm{HalfNormal}(1, 0.25) \\
\rho &\sim \mathrm{HalfNormal}(3, 0.25)
\end{aligned}
$$

Well lets think about missing values, we could make models for them and then use them to impute the missing values. For instance this kind of model somewhat implies a model for G

$$
\begin{aligned}
G &\sim \mathrm{MVNormal}(\nu_i, K_G) \\
\nu_i &= \alpha_G + \beta_{G,M} M_i \\
K &= \eta_G^2 \exp(-\rho_G d_{ij}) \\
\alpha_G &\sim \mathrm{Normal}(0, 1) \\
\beta_{G,M} &\sim \mathrm{Normal}(0, 0.5) \\
\eta_G^2 &\sim \mathrm{HalfNormal}(1, 0.25) \\
\rho_G &\sim \mathrm{HalfNormal}(3, 0.25)
\end{aligned}
$$

and similar for M

$$
\begin{aligned}
M &\sim \text{MVNormal}(0, K_M) \\
K_M &= \eta_M^2 \exp(-\rho_M d_{ij}) \\
\alpha_M &\sim \text{Normal}(0, 1) \\
\eta_M^2 &\sim \text{HalfNormal}(1, 0.25) \\
\rho_M &\sim \text{HalfNormal}(3, 0.25).
\end{aligned}
$$

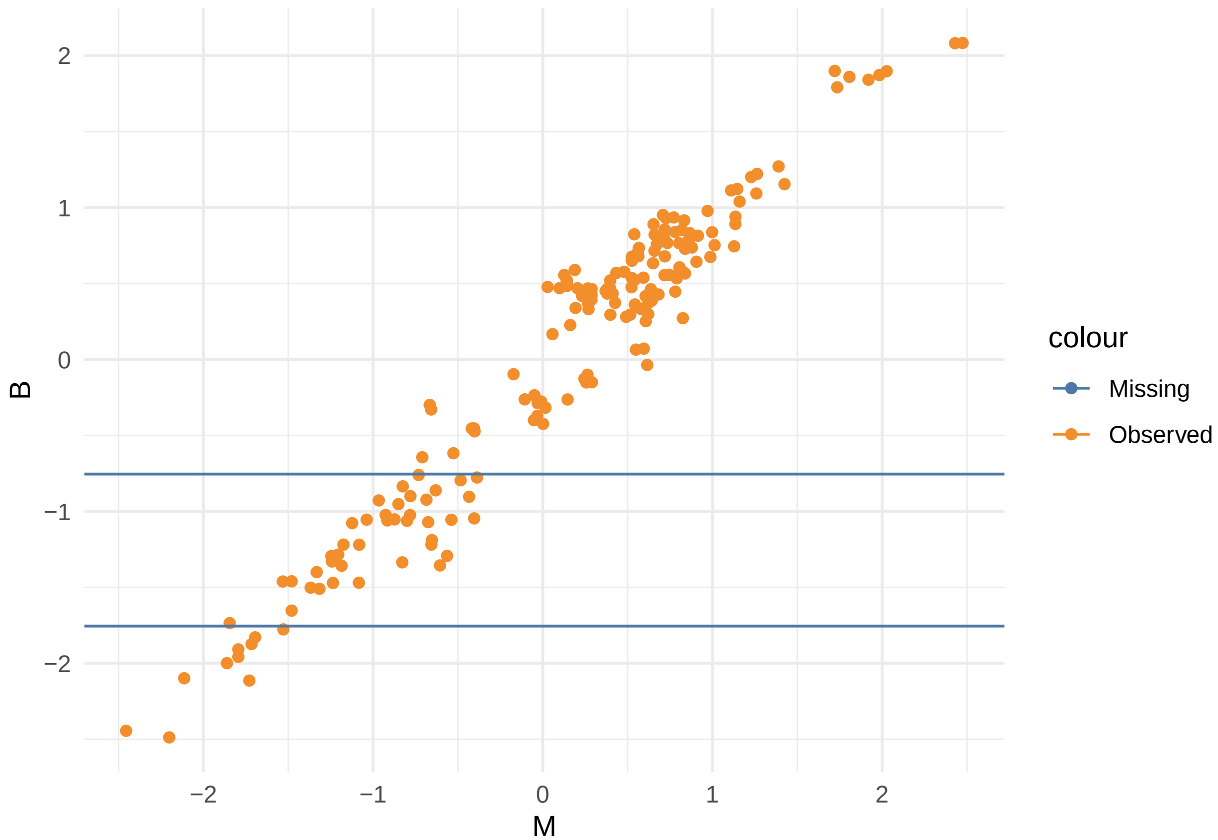
We really haven't done much here, except follow the consequences naturally implied by our first model; if our sub model relationships where different, we would expect different relationships for the overarching model. STAN can run these three models simultaneously, cascading implications across them. Consider this in comparison to the independent missing value model

$$
\begin{aligned}
G &\sim \text{Normal}(0, 1) \\
M &\sim \text{Normal}(0, 1).
\end{aligned}
$$

```
data(Primates301)
data(Primates301_nex)
df <- Primates301 %>%
    mutate( G = standardize(log(group_size))
          , M = standardize(log(body))
          , B = standardize(log(brain))
          , name = as.character(name) ) %>%
    subset( complete.cases(B) )

dfc <-  subset( df, complete.cases(B,M,G))


names        <- df$name
tree_trimmed <- keep.tip(Primates301_nex, names)
Dmat         <- cophenetic(tree_trimmed)

d <- list( G           = ifelse(is.na(df$G),-99,df$G)
         , M           = ifelse(is.na(df$M),-99,df$M)
         , B           = ifelse(is.na(df$B),-99,df$B)
         , N_G_miss    = sum(is.na(df$G))
         , N_M_miss    = sum(is.na(df$M))
         , N_B_miss    = sum(is.na(df$B))
         , G_miss_idx = which(is.na(df$G))
         , M_miss_idx = which(is.na(df$M))
         , B_miss_idx = which(is.na(df$B))
         , N           = nrow(df)
         , Dmat        = Dmat[names,names] / max(Dmat))

d2 <- list( G           = df$G
          , M           = df$M
```

```r
          , B           = df$B
          , N_G_miss    = sum(is.na(df$G))
          , N_M_miss    = sum(is.na(df$M))
          , N_B_miss    = sum(is.na(df$B))
          , G_miss_idx = which(is.na(df$G))
          , M_miss_idx = which(is.na(df$M))
          , B_miss_idx = which(is.na(df$B))
          , N           = nrow(df)
          , Dmat        = Dmat[names,names] / max(Dmat))

namesc        <- dfc$name
tree_trimmedc <- keep.tip(Primates301_nex, namesc)
Dmatc         <- cophenetic(tree_trimmedc)
dc <- list( G           = dfc$G
          , M           = dfc$M
          , B           = dfc$B
          , N           = nrow(dfc)
          , Dmat        = Dmatc[namesc,namesc] / max(Dmatc))
```

```r
ggplot(df) +
    geom_point(aes(x=M, y=B, colour='Observed'), na.rm=T) +
    geom_hline(aes(yintercept=B, colour='Missing'), data=df %>% subset(is.na(M)), na.rm=T)
    scale_color_tableau() +
    theme_minimal()
```



So of course lets try the simple model first

```
m1 <- cstan(file='../models/l18_minimal.stan', data=d, chains=4, cores=12, threads=3, iter=
```

```
## Running MCMC with 4 chains, at most 12 in parallel, with 3 thread(s) per chain...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 1 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 1 but if this warning occurs often then your model may be either severely ill-cond
## Chain 1
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 1 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 1 but if this warning occurs often then your model may be either severely ill-cond
## Chain 1
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b
```

```
## Chain 1 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 1 but if this warning occurs often then your model may be either severely ill-cond
## Chain 1
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 2 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat
## Chain 2 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 2 but if this warning occurs often then your model may be either severely ill-cond
## Chain 2
## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 2 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat
## Chain 2 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 2 but if this warning occurs often then your model may be either severely ill-cond
## Chain 2
## Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 4 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 4 but if this warning occurs often then your model may be either severely ill-cond
## Chain 4
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 4 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 4 but if this warning occurs often then your model may be either severely ill-cond
## Chain 4
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 4 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 4 but if this warning occurs often then your model may be either severely ill-cond
```

```
## Chain 4
```

```
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 4 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 3 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 4 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 3 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 4 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 3 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 4 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 3 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 3 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 4 Iteration: 1200 / 2000 [ 60%]  (Sampling)
```

```
## Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 4 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 3 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 4 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 4 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 129.9 seconds.
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4 finished in 135.8 seconds.
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 3 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3 finished in 143.2 seconds.
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 144.9 seconds.
##
## All 4 chains finished successfully.
```

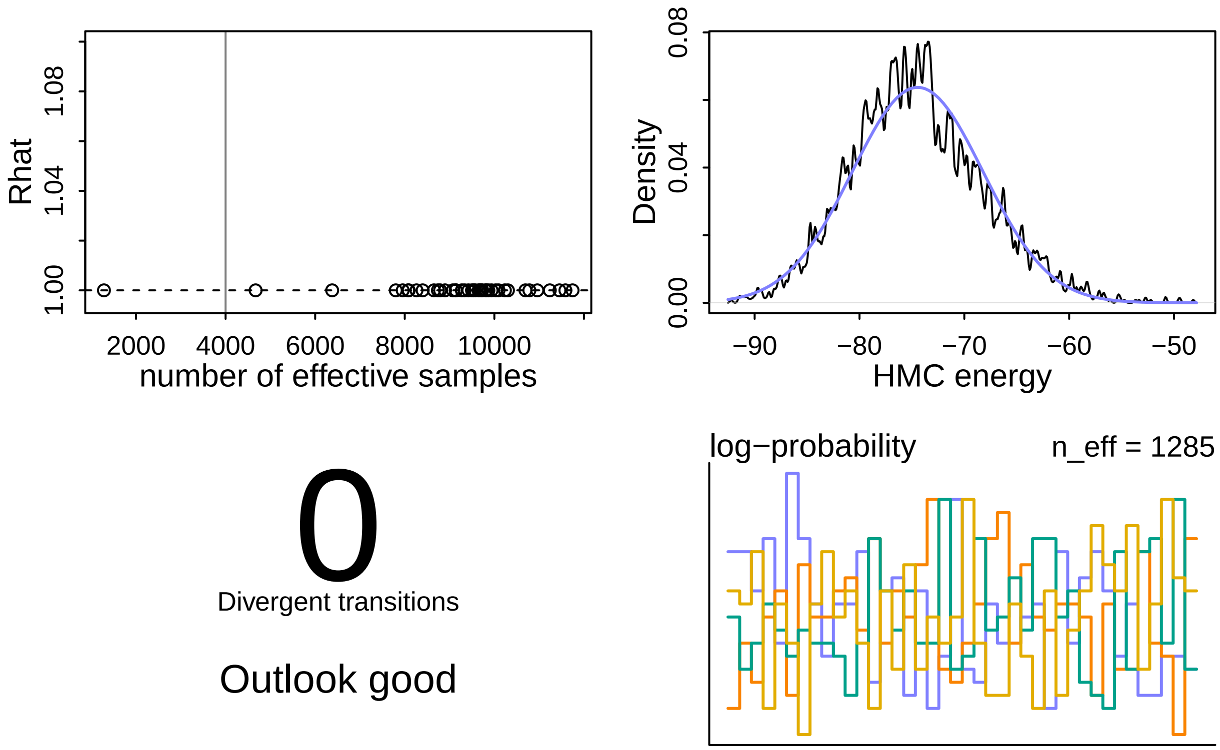
```
## Mean chain execution time: 138.5 seconds.
## Total execution time: 145.0 seconds.
```

```r
precis(m1, depth=2)
```

```
##                      mean          sd        5.5%       94.5%     n_eff
## M_impute[1]  -1.574320322 0.161727910 -1.83499990 -1.31969735  7953.963
## M_impute[2]  -0.550302528 0.149008202 -0.79033159 -0.31715005  9649.573
## G_impute[1]  -0.003651525 0.969017598 -1.54152970  1.54524280  8789.957
## G_impute[2]   0.028958369 0.980571604 -1.51920290  1.61282135  9421.213
## G_impute[3]   0.004758838 0.963544063 -1.53047845  1.57463950  9138.460
## G_impute[4]   0.052432207 0.959546397 -1.48487325  1.60663205 11237.699
## G_impute[5]   0.083786968 1.034176235 -1.55004180  1.74814355 10782.111
## G_impute[6]   0.147896274 1.035317705 -1.51111075  1.83265015  9274.630
## G_impute[7]   0.055848379 0.979574691 -1.51553550  1.62540535 10693.966
## G_impute[8]   0.116926825 1.008501284 -1.50456520  1.74776540  8262.159
## G_impute[9]   0.024548655 1.027620733 -1.60572725  1.64378850  9729.313
## G_impute[10]  0.029302722 0.962419409 -1.53612675  1.52683475  8399.529
## G_impute[11]  0.052876128 0.990203814 -1.54394360  1.64473810  9936.771
## G_impute[12]  0.072399495 0.974377316 -1.50623030  1.66028540  8739.985
## G_impute[13] -0.038396410 0.963627931 -1.54201030  1.48277495  9571.311
## G_impute[14]  0.007133839 0.983128697 -1.56916865  1.57215660 11452.334
## G_impute[15] -0.127158037 0.990952570 -1.75337220  1.46288695  9138.980
## G_impute[16] -0.054338055 0.982408230 -1.61053000  1.49332170  9811.828
## G_impute[17] -0.014553866 0.991306765 -1.59366865  1.54574510  9335.941
## G_impute[18] -0.059717290 0.999720092 -1.65263225  1.57367090  8650.840
## G_impute[19] -0.152781241 1.006188089 -1.77667915  1.47674030  8088.293
## G_impute[20]  0.003439061 0.984353388 -1.58483880  1.58081600  9075.222
## G_impute[21] -0.070348682 0.983632372 -1.62918725  1.49112115  9708.055
## G_impute[22] -0.047247843 0.993696284 -1.65340750  1.53839140 11586.738
## G_impute[23]  0.088483077 1.025650156 -1.54914380  1.73301845  9812.831
## G_impute[24] -0.039764735 0.993867122 -1.63436405  1.52034830 10092.805
## G_impute[25] -0.039293450 0.999437022 -1.63152875  1.56428720  8074.699
## G_impute[26]  0.181098645 1.011813293 -1.46723260  1.79179615  8898.797
## G_impute[27]  0.079854638 1.010849877 -1.53361235  1.65919180 11745.744
## G_impute[28] -0.037318958 1.005477504 -1.64595650  1.58179880 10303.192
## G_impute[29]  0.021713593 0.996747113 -1.56248020  1.58543510 10030.273
## G_impute[30] -0.002622461 0.971147216 -1.53328475  1.55502990 10959.242
## G_impute[31]  0.040478478 1.024195543 -1.60042480  1.66779015  9502.133
## G_impute[32]  0.068459131 1.017194281 -1.56757480  1.69739630 10234.345
## G_impute[33] -0.019751459 0.988432147 -1.60608310  1.60179675  9537.714
## alpha_B      -0.058063788 0.879264882 -1.44326250  1.33899520  9866.777
## betaB_G       0.010495009 0.017226078 -0.01635760  0.03826006  4670.230
## betaB_M       0.823736645 0.032156277  0.77150501  0.87426248  9678.013
## etasqB        2.968906295 0.252001154  2.56581575  3.37138825  7799.114
## rhoB          0.021600216 0.004504822  0.01527379  0.02939212  6379.371
```

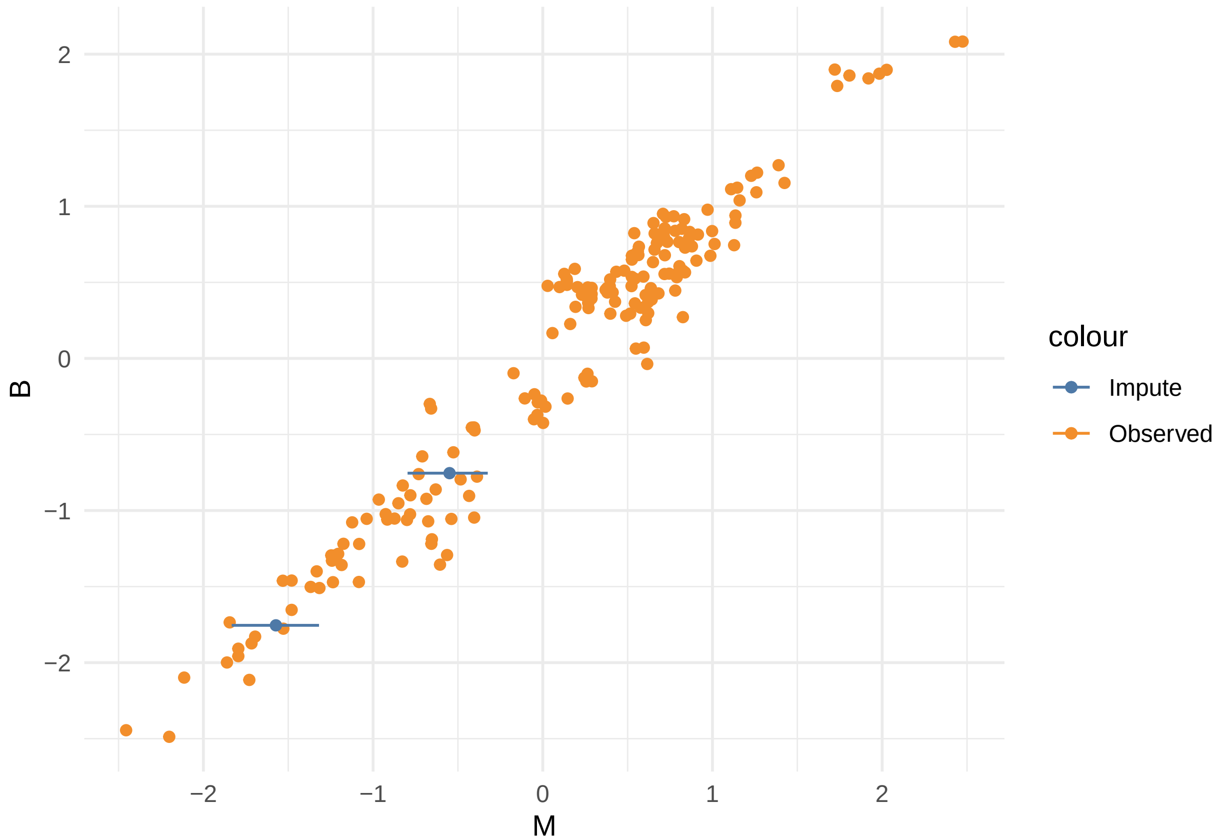
```
##                  Rhat4
## M_impute[1]  0.9993451
## M_impute[2]  0.9992232
## G_impute[1]  0.9993378
## G_impute[2]  0.9992042
## G_impute[3]  0.9997537
## G_impute[4]  0.9992287
## G_impute[5]  0.9993150
## G_impute[6]  0.9993644
## G_impute[7]  0.9994987
## G_impute[8]  0.9993774
## G_impute[9]  0.9992175
## G_impute[10] 0.9999340
## G_impute[11] 0.9994685
## G_impute[12] 0.9993603
## G_impute[13] 0.9990704
## G_impute[14] 0.9992173
## G_impute[15] 0.9997831
## G_impute[16] 0.9991282
## G_impute[17] 0.9996440
## G_impute[18] 0.9994783
## G_impute[19] 1.0000754
## G_impute[20] 0.9992778
## G_impute[21] 0.9991037
## G_impute[22] 0.9996383
## G_impute[23] 0.9992081
## G_impute[24] 0.9993063
## G_impute[25] 0.9994668
## G_impute[26] 0.9992457
## G_impute[27] 0.9991457
## G_impute[28] 0.9998979
## G_impute[29] 0.9991850
## G_impute[30] 0.9991896
## G_impute[31] 0.9995469
## G_impute[32] 0.9991255
## G_impute[33] 0.9997882
## alpha_B      0.9993494
## betaB_G      0.9997164
## betaB_M      0.9992934
## etasqB       0.9993247
## rhoB         0.9999976
```

```
dashboard(m1)
```

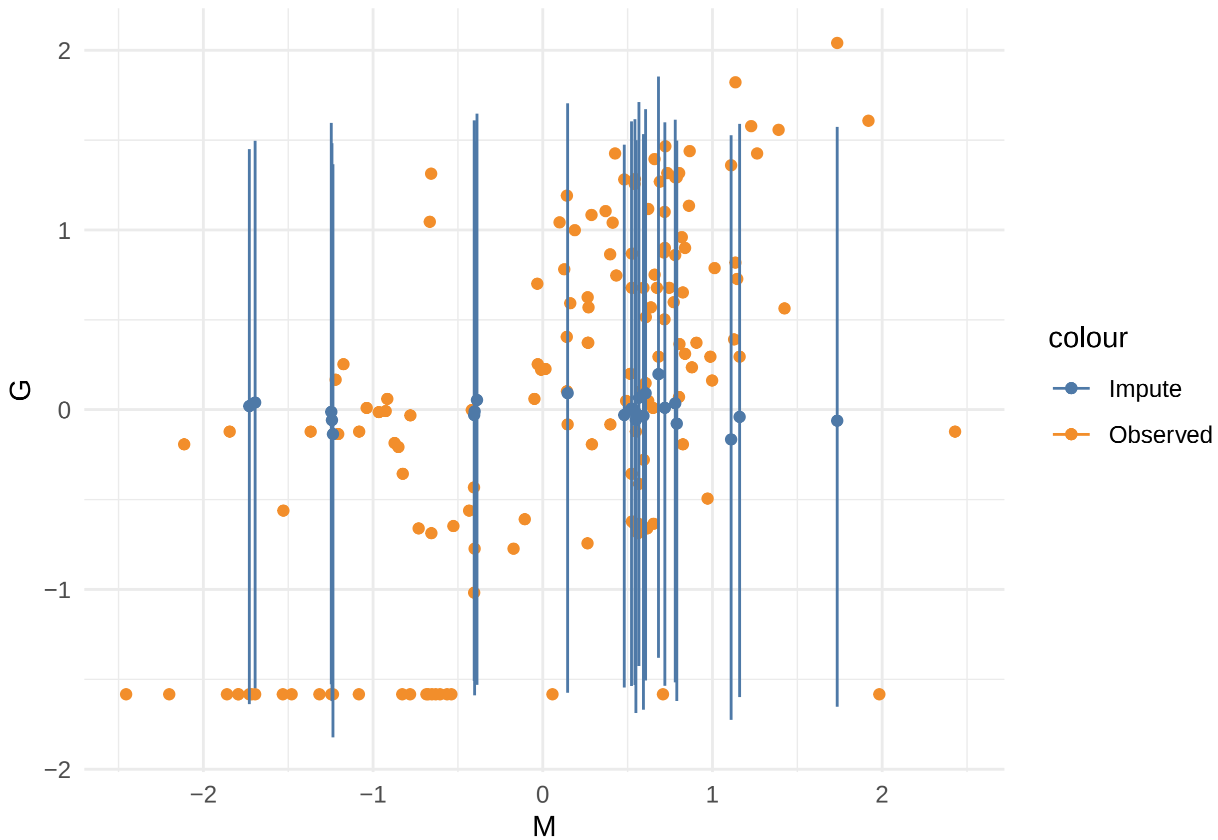



```
p1 <- extract.samples(m1)

gen_df <- function(p){
    df1_M <- p$M_impute %>%
        apply(2, post_summary) %>%
        as.data.frame %>%
        pivot_longer(everything(), names_to='M', names_prefix='V', values_to='M_impute') %>
        group_by(M) %>% summarise(M_impute_lower=min(M_impute), M_impute_median=median(M_im
        mutate(M = df[d$M_miss_idx,] %>% rownames()) %>%
        column_to_rownames('M')

    df1_G <- p$G_impute %>%
        apply(2, post_summary) %>%
        as.data.frame %>%
        pivot_longer(everything(), names_to='G', names_prefix='V', values_to='G_impute') %>
        group_by(G) %>% summarise(G_impute_lower=min(G_impute), G_impute_median=median(G_im
        mutate(G = df[d$G_miss_idx,] %>% rownames()) %>%
        column_to_rownames('G')

    df1 <- df %>%
        merge(df1_M, by='row.names', all.x=T) %>%
        select(-'Row.names') %>%
        merge(df1_G, by='row.names', all.x=T) %>%
        select(-'Row.names')
}
```

```r
df1 <- gen_df(p1)

ggplot(df1) +
    geom_point(aes(x=M, y=B, colour='Observed'), na.rm=T) +
    geom_segment(aes(x=M_impute_lower, xend=M_impute_upper, y=B, yend=B, colour="Impute"),
    geom_point(  aes(x=M_impute_median                    , y=B         , colour="Impute"),
    scale_color_tableau() +
    theme_minimal()
```




```r
ggplot(df1) +
    geom_point(aes(x=M, y=G, colour='Observed'), na.rm=T) +
    geom_segment(aes(x=M, xend=M, y=G_impute_lower, yend=G_impute_upper, colour="Impute"),
    geom_point(  aes(x=M          , y=G_impute_median                    , colour="Impute"),
    scale_color_tableau() +
    theme_minimal()
```



As we can see from our imputation, the model requires no phylogeny to impute M, however it does not impute a precise G at all.

```
m2a <- cstan(file='../models/l18_BG_phylo.stan', data=d, chains=4, cores=12, threads=3, ite
```

```
## Warning in readLines(stan_file): incomplete final line found on '../models/
## l18_BG_phylo.stan'

## Running MCMC with 4 chains, at most 12 in parallel, with 3 thread(s) per chain...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)

## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 2 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat

## Chain 2 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 2 but if this warning occurs often then your model may be either severely ill-cond

## Chain 2

## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 2 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat

## Chain 2 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 2 but if this warning occurs often then your model may be either severely ill-cond
```

```
## Chain 2
## Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 3 Exception: multi_normal_lpdf: LDLT_Factor of covariance parameter is not positiv
## Chain 3 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 3 but if this warning occurs often then your model may be either severely ill-cond
## Chain 3
## Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 1 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 1 but if this warning occurs often then your model may be either severely ill-cond
## Chain 1
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 4 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 4 but if this warning occurs often then your model may be either severely ill-cond
## Chain 4
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 4 Exception: multi_normal_lpdf: LDLT_Factor of covariance parameter is not positiv
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 4 but if this warning occurs often then your model may be either severely ill-cond
## Chain 4
## Chain 4 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 3 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 4 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 4 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4 Iteration:  400 / 2000 [ 20%]  (Warmup)
```

```
## Chain 3 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 4 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 3 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 4 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 3 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 4 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 3 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 4 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 4 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 3 Iteration: 1400 / 2000 [ 70%]  (Sampling)
```

```
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 4 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 3 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 4 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4 finished in 192.1 seconds.
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3 finished in 197.2 seconds.
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 201.6 seconds.
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 203.8 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 198.7 seconds.
## Total execution time: 204.0 seconds.
```

```r
m2b <- cstan(file='../models/l18_BG_model.stan', data=d, chains=4, cores=12, threads=3, ite
```

```
## Warning in readLines(stan_file): incomplete final line found on '../models/
## l18_BG_model.stan'
```

```
## Running MCMC with 4 chains, at most 12 in parallel, with 3 thread(s) per chain...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
```

```
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b
```

```
## Chain 1 Exception: multi_normal_lpdf: LDLT_Factor of covariance parameter is not positiv
```

```
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 1 but if this warning occurs often then your model may be either severely ill-cond
## Chain 1
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 1 Exception: multi_normal_lpdf: LDLT_Factor of covariance parameter is not positiv
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 1 but if this warning occurs often then your model may be either severely ill-cond
## Chain 1
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 1 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 1 but if this warning occurs often then your model may be either severely ill-cond
## Chain 1
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 1 Exception: multi_normal_lpdf: LDLT_Factor of covariance parameter is not positiv
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 1 but if this warning occurs often then your model may be either severely ill-cond
## Chain 1
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 2 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/tmp/Rt
## Chain 2 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 2 but if this warning occurs often then your model may be either severely ill-cond
## Chain 2
## Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 4 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 4 but if this warning occurs often then your model may be either severely ill-cond
## Chain 4
```

```
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 4 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat

## Chain 4 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 4 but if this warning occurs often then your model may be either severely ill-cond

## Chain 4

## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 4 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/tmp/Rt

## Chain 4 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 4 but if this warning occurs often then your model may be either severely ill-cond

## Chain 4

## Chain 4 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 4 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 3 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 4 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 3 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 4 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 4 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4 Iteration:  900 / 2000 [ 45%]  (Warmup)
```

```
## Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 3 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 4 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 4 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 3 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 4 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 4 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3 Iteration: 1700 / 2000 [ 85%]  (Sampling)
```

```
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 99.1 seconds.
## Chain 3 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4 finished in 100.3 seconds.
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 101.0 seconds.
## Chain 3 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3 finished in 105.5 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 101.5 seconds.
## Total execution time: 105.6 seconds.
```

```
m2 <- cstan(file='../models/l18_BG.stan'      , data=d, chains=4, cores=12, threads=3, ite
```

```
## Warning in readLines(stan_file): incomplete final line found on '../models/
## l18_BG.stan'
```

```
## Running MCMC with 4 chains, at most 12 in parallel, with 3 thread(s) per chain...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)

## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 1 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat

## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 1 but if this warning occurs often then your model may be either severely ill-cond

## Chain 1

## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)

## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 2 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat

## Chain 2 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 2 but if this warning occurs often then your model may be either severely ill-cond

## Chain 2

## Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)

## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 3 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat

## Chain 3 If this warning occurs sporadically, such as for highly constrained variable typ
```

```
## Chain 3 but if this warning occurs often then your model may be either severely ill-cond

## Chain 3

## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 3 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat

## Chain 3 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 3 but if this warning occurs often then your model may be either severely ill-cond

## Chain 3

## Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 4 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 3 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 4 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 4 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 4 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 3 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 4 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 3 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 4 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 3 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 4 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 3 Iteration:  900 / 2000 [ 45%]  (Warmup)
```

```
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 4 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 4 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 3 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 4 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 3 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 4 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 163.9 seconds.
## Chain 3 Iteration: 1900 / 2000 [ 95%]  (Sampling)
```

```
## Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4 finished in 164.7 seconds.
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3 finished in 169.9 seconds.
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 175.6 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 168.5 seconds.
## Total execution time: 175.7 seconds.
```

```r
precis(m2a, depth=2)
```

```
##                      mean         sd        5.5%       94.5%     n_eff     Rhat4
## M_impute[1]   -1.54676105 0.16339168 -1.81085495 -1.28304175  7687.909 0.9997896
## M_impute[2]   -0.53325206 0.14158796 -0.76832570 -0.30632397  7757.998 0.9998263
## G_impute[1]    0.14602230 0.34320813 -0.40367357  0.70249532  9474.010 0.9993504
## G_impute[2]   -0.60808953 0.19134032 -0.91129607 -0.30892367  7233.480 0.9991425
## G_impute[3]   -1.55938111 0.28671776 -2.01480660 -1.10444270  8177.618 0.9992814
## G_impute[4]    1.09107699 0.30423763  0.60122620  1.57737265  8195.377 0.9996647
## G_impute[5]    0.37698115 0.44005345 -0.31739160  1.07660000  9494.482 0.9999056
## G_impute[6]    0.92349936 0.25272042  0.51542728  1.32751045  7021.093 0.9993888
## G_impute[7]    0.40038492 0.19801564  0.08533383  0.71075504  7312.110 1.0000232
## G_impute[8]    0.62963512 0.27937214  0.18131863  1.07276605  7609.924 0.9996976
## G_impute[9]    0.68607150 0.54859499 -0.18333465  1.55553870  8017.119 0.9993470
## G_impute[10]   0.50782198 0.36211851 -0.07596629  1.10031080  7826.189 0.9995902
## G_impute[11]   0.60332367 0.32834672  0.07099681  1.13190935  7861.874 0.9994025
## G_impute[12]   0.89213313 0.58956113 -0.05287128  1.81370940  2989.774 1.0007951
## G_impute[13]   0.85844200 0.58448046 -0.07284438  1.78374815  3220.108 0.9997740
## G_impute[14]   0.85911167 0.58709286 -0.05752259  1.79981645  3211.440 1.0007510
## G_impute[15]   0.07587712 0.26159974 -0.34960139  0.49893713  7907.670 0.9995748
## G_impute[16]  -0.03167212 0.34584513 -0.56882327  0.51900171  7232.702 0.9992421
## G_impute[17]  -1.49812505 0.37645379 -2.09918925 -0.88864185  7948.846 0.9994853
## G_impute[18]   0.01907607 0.41518750 -0.63095966  0.68343989  7968.274 0.9992490
## G_impute[19]  -0.61501501 0.31127604 -1.10008080 -0.12059398  6725.511 0.9995661
## G_impute[20]  -0.60516706 0.26881915 -1.03725165 -0.17787141  7285.316 0.9992830
## G_impute[21]  -1.58767202 0.26530754 -2.01444990 -1.16974725 10069.437 0.9994192
## G_impute[22]   1.00216027 0.44319493  0.29951043  1.70589135  8599.632 0.9996231
## G_impute[23]   1.05870511 0.37874493  0.45220784  1.66786930  7253.249 1.0003296
## G_impute[24]  -1.58809699 0.21441664 -1.93164880 -1.24750610  7240.247 0.9992269
## G_impute[25]  -1.58748373 0.38248890 -2.19552840 -0.98259009  6924.993 0.9992775
## G_impute[26]   1.62693002 0.20144678  1.30780360  1.94857430  6542.056 1.0002410
## G_impute[27]   1.62094447 0.27378170  1.17792740  2.05563590  6853.374 0.9998220
## G_impute[28]   1.36387256 0.28954601  0.90640392  1.83151825  7201.306 0.9994433
```

```
## G_impute[29] -1.03865231 0.93044032 -2.53603665  0.43627247  7704.952 1.0000379
## G_impute[30] -1.14859036 0.51593722 -1.96262220 -0.33044812  8328.026 0.9995367
## G_impute[31] -0.10012543 0.23520273 -0.47673950  0.28194977  7177.164 0.9999285
## G_impute[32] -0.10344718 0.37305603 -0.69816260  0.49087467  7307.522 0.9992884
## G_impute[33] -0.72365191 0.32048467 -1.25019640 -0.21870683  8199.524 0.9995572
## alpha_B      -0.05508634 0.86166784 -1.44400665  1.29292980 11044.001 1.0000019
## betaB_G       0.04603696 0.02214522  0.01133884  0.08088704  6778.385 0.9998111
## betaB_M       0.81689818 0.03080700  0.76746150  0.86583827  7914.721 0.9994747
## etasqG        2.89889655 0.23828474  2.52529295  3.27980550  5307.290 0.9994831
## rhoG          0.86567919 0.12944614  0.67218901  1.08939330  3845.290 0.9995776
## etasqB        2.96583802 0.24873302  2.57186875  3.36394410  6668.841 0.9995856
## rhoB          0.02042180 0.00426081  0.01432279  0.02766432  5322.621 0.9996167
```

```
precis(m2b, depth=2)
```

```
##                      mean          sd        5.5%       94.5%     n_eff
## M_impute[1]  -1.565696430 0.159604217 -1.81865540 -1.30891570  8479.275
## M_impute[2]  -0.543588165 0.143779500 -0.77205392 -0.31317745  6973.228
## G_impute[1]   0.427181625 0.782928421 -0.86578516  1.68401935 10035.539
## G_impute[2]  -0.434045188 0.780947422 -1.72584545  0.80170885  8889.310
## G_impute[3]  -0.942241363 0.750328479 -2.14081760  0.25102790  7082.445
## G_impute[4]   0.586859363 0.788200018 -0.67338408  1.81951530  8940.699
## G_impute[5]   0.092569432 0.762244735 -1.12132255  1.25337990  9731.707
## G_impute[6]   0.579908900 0.785033162 -0.66032243  1.82154430  8535.734
## G_impute[7]   0.134486430 0.780157238 -1.11299325  1.39537885  9029.993
## G_impute[8]   0.238471559 0.762348542 -0.97857612  1.42568950  8503.064
## G_impute[9]   0.274699652 0.756284247 -0.93174730  1.48041485  8555.073
## G_impute[10]  0.213888144 0.780393303 -1.05065660  1.46057710  7606.051
## G_impute[11]  0.190882746 0.784648541 -1.07717430  1.42742765  8398.684
## G_impute[12]  0.249173233 0.745213295 -0.89887746  1.42656695  7601.065
## G_impute[13]  0.231957609 0.727083053 -0.92934853  1.39641025  9209.308
## G_impute[14]  0.254053542 0.750639883 -0.95184501  1.45969790  7238.906
## G_impute[15] -0.101664202 0.776698630 -1.34383085  1.13381030  8849.044
## G_impute[16] -0.092202534 0.744496741 -1.26815550  1.10601335  7024.192
## G_impute[17] -1.035695273 0.784994423 -2.28120790  0.24470950  8374.796
## G_impute[18]  1.515772978 0.790867707  0.25441906  2.79949085  7340.020
## G_impute[19] -0.390904477 0.755756311 -1.58435825  0.81869654  9220.706
## G_impute[20] -0.348271555 0.765657379 -1.57825055  0.86755661  8368.894
## G_impute[21] -0.909914597 0.771645542 -2.12426530  0.31614824  7305.208
## G_impute[22]  0.554257590 0.765266830 -0.69264477  1.76436490  8770.022
## G_impute[23]  0.425958898 0.753542581 -0.80064243  1.59112880  7718.548
## G_impute[24] -0.347052311 0.762243737 -1.55306360  0.89009894  9146.497
## G_impute[25] -0.543784727 0.778665608 -1.79381025  0.69176479  9116.508
## G_impute[26]  1.245371262 0.773223288  0.01643200  2.44931195  7548.739
## G_impute[27]  1.220801614 0.774190396 -0.01152191  2.43513200  8722.313
## G_impute[28]  0.748504751 0.772580369 -0.46726836  1.98796770  7470.970
```

```
## G_impute[29] -0.744668455 0.779839389 -1.97394845  0.49094455  7705.865
## G_impute[30]  1.288761631 0.778406451  0.02549319  2.51299220  9352.566
## G_impute[31]  0.184650940 0.764674186 -1.02144610  1.36977055  7970.422
## G_impute[32] -0.653069121 0.782156573 -1.86860070  0.65146507  7939.922
## G_impute[33]  0.143577700 0.783754784 -1.10600815  1.37614215  9906.223
## alpha_G      -0.001401478 0.062710638 -0.10170227  0.09995195  6714.297
## alpha_B      -0.050002622 0.873295472 -1.43893165  1.31302790  9309.805
## betaG_M       0.651178820 0.066440789  0.54154645  0.75460787  7412.821
## betaB_G       0.016011020 0.019577832 -0.01499608  0.04710893  5203.113
## betaB_M       0.820322731 0.032363121  0.76853343  0.87150706  7239.834
## sigma_G       0.769307668 0.044997319  0.70227355  0.84417007  5099.267
## etasqB        2.963295800 0.246773051  2.57499285  3.34445585  6748.713
## rhoB          0.021483182 0.004386495  0.01540367  0.02905581  5791.829
##                  Rhat4
## M_impute[1]  0.9992227
## M_impute[2]  0.9992964
## G_impute[1]  0.9993676
## G_impute[2]  0.9991587
## G_impute[3]  0.9996444
## G_impute[4]  0.9997691
## G_impute[5]  0.9991298
## G_impute[6]  0.9994420
## G_impute[7]  0.9998596
## G_impute[8]  0.9993288
## G_impute[9]  1.0002020
## G_impute[10] 0.9994181
## G_impute[11] 0.9995041
## G_impute[12] 0.9996006
## G_impute[13] 0.9992941
## G_impute[14] 0.9991169
## G_impute[15] 0.9993127
## G_impute[16] 0.9993104
## G_impute[17] 0.9991071
## G_impute[18] 0.9998817
## G_impute[19] 0.9994615
## G_impute[20] 0.9995884
## G_impute[21] 0.9996588
## G_impute[22] 0.9996796
## G_impute[23] 0.9994762
## G_impute[24] 0.9992082
## G_impute[25] 0.9994016
## G_impute[26] 0.9993721
## G_impute[27] 0.9993081
## G_impute[28] 0.9995326
## G_impute[29] 0.9995148
## G_impute[30] 0.9993845
```

```
## G_impute[31] 0.9996366
## G_impute[32] 0.9999470
## G_impute[33] 0.9991031
## alpha_G      0.9994970
## alpha_B      0.9994912
## betaG_M      0.9998168
## betaB_G      0.9995570
## betaB_M      1.0002035
## sigma_G      0.9993293
## etasqB       1.0001557
## rhoB         0.9999404
```

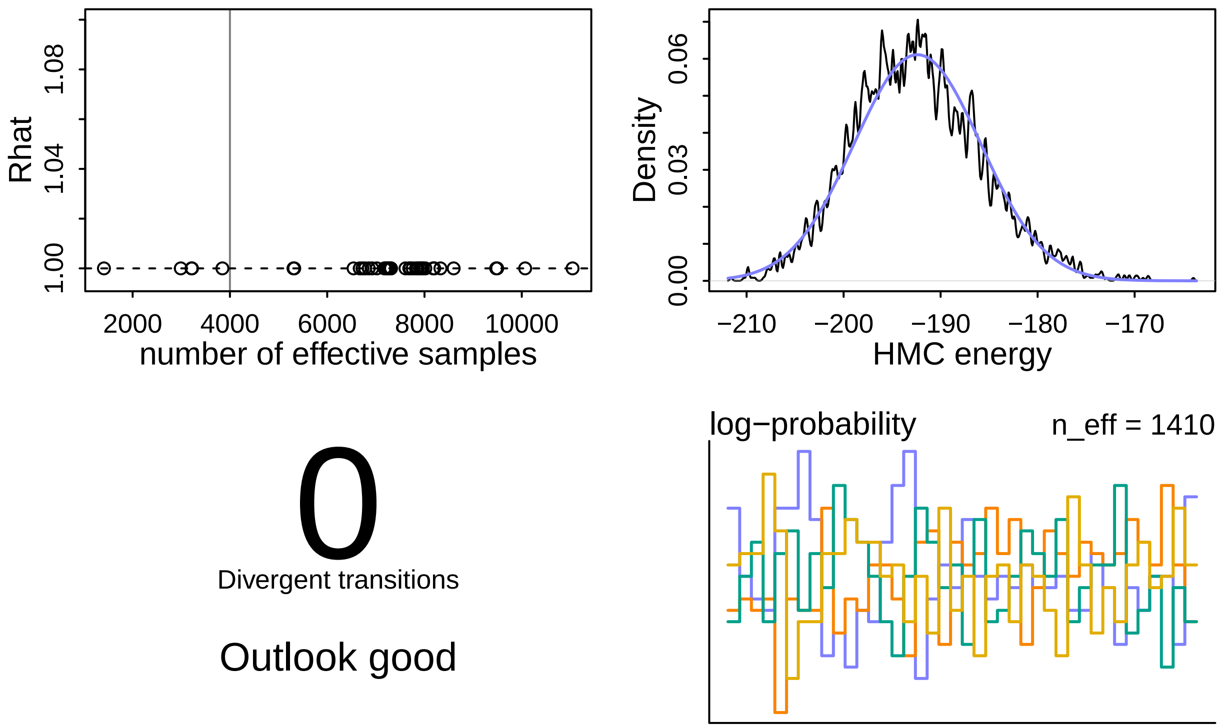
```
precis(m2 , depth=2)
```

```
##                      mean          sd         5.5%       94.5%     n_eff
## M_impute[1]   -1.54931135 0.161285256 -1.802999350 -1.28937945 10011.053
## M_impute[2]   -0.53665886 0.147188323 -0.772123140 -0.29953720  7072.764
## G_impute[1]    0.16719987 0.338958317 -0.387101240  0.71347622  9115.683
## G_impute[2]   -0.66317068 0.195817226 -0.970570195 -0.35144052  7731.319
## G_impute[3]   -1.62672407 0.284420447 -2.070074250 -1.16177580  7957.452
## G_impute[4]    1.12092445 0.302276915  0.640932835  1.60596805 10400.802
## G_impute[5]    0.35890523 0.448687791 -0.364946055  1.08705180  9955.803
## G_impute[6]    0.91269018 0.253257791  0.512118240  1.32654200  8157.916
## G_impute[7]    0.38183823 0.185442600  0.082384180  0.68833716  7503.384
## G_impute[8]    0.59500272 0.273441675  0.163388710  1.02881045  9918.097
## G_impute[9]    0.64068740 0.526612836 -0.204452160  1.47743265  7814.268
## G_impute[10]   0.49218721 0.354483681 -0.064181305  1.06881235  8799.286
## G_impute[11]   0.56919154 0.333648694  0.044820843  1.09651925  7882.098
## G_impute[12]   0.80091220 0.578274578 -0.126523555  1.71363660  2758.298
## G_impute[13]   0.80047568 0.572494828 -0.087041665  1.73386040  2933.774
## G_impute[14]   0.79667800 0.572798526 -0.120641695  1.73633695  3546.323
## G_impute[15]   0.09253433 0.257729945 -0.322975405  0.50502292  6676.140
## G_impute[16]  -0.01379781 0.343021972 -0.566783180  0.53839496  8141.413
## G_impute[17]  -1.47583833 0.377995023 -2.073354300 -0.87404352  9774.003
## G_impute[18]   0.09430361 0.411118647 -0.560218555  0.74682032  6842.906
## G_impute[19]  -0.53573070 0.313336589 -1.036472100 -0.03736301  6473.729
## G_impute[20]  -0.56543041 0.266484876 -0.989494055 -0.14208135  7140.164
## G_impute[21]  -1.53264808 0.259660359 -1.943544300 -1.11780790  8428.519
## G_impute[22]   1.07872928 0.443212484  0.379958855  1.76672945  7101.525
## G_impute[23]   0.99955198 0.390746402  0.375938455  1.62713200  8300.129
## G_impute[24]  -1.49927338 0.218675669 -1.843557650 -1.14169725  7463.887
## G_impute[25]  -1.57336123 0.387120780 -2.191021450 -0.96310731  7103.694
## G_impute[26]   1.57641869 0.199716730  1.257004700  1.90237265  6226.182
## G_impute[27]   1.59390350 0.286062454  1.147768350  2.05819330  8120.400
## G_impute[28]   1.38279968 0.298986314  0.905277555  1.85462550  8502.922
## G_impute[29]  -1.15410357 0.915264999 -2.584632050  0.33481243  8197.336
```

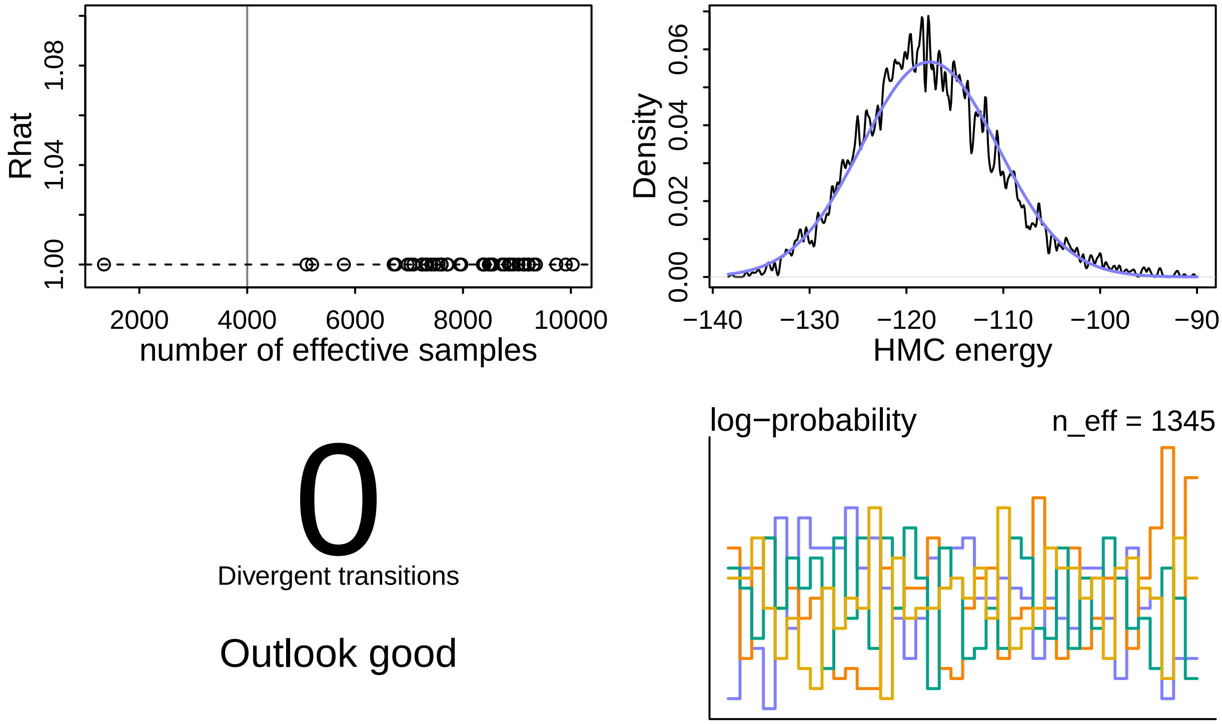
```
## G_impute[30] -1.11321656 0.486504781 -1.878538150 -0.34608970  7825.542
## G_impute[31] -0.07821436 0.242596625 -0.460611010  0.31372712 10002.334
## G_impute[32] -0.18370650 0.373384602 -0.792543420  0.41091887  7836.508
## G_impute[33] -0.71286586 0.304928372 -1.191018700 -0.22379777  8296.138
## alpha_G      -0.25651299 0.797051414 -1.516090950  1.02056150  8313.097
## alpha_B      -0.05725328 0.890169894 -1.476377750  1.35424605  8934.268
## betaG_M       0.27551092 0.141417144  0.048897620  0.50040698  6547.324
## betaB_G       0.04319879 0.022335719  0.007206378  0.07856378  6554.157
## betaB_M       0.81679936 0.032156839  0.764374865  0.86588427  7586.580
## etasqG        2.89265875 0.241253441  2.511758100  3.28012760  6426.234
## rhoG          0.84951982 0.130111174  0.656229985  1.07329870  4009.306
## etasqB        2.95955198 0.254237007  2.561803250  3.36757650  6462.219
## rhoB          0.02056521 0.004326705  0.014373073  0.02821599  5327.380
##                  Rhat4
## M_impute[1]  0.9992068
## M_impute[2]  0.9999851
## G_impute[1]  0.9996726
## G_impute[2]  0.9995156
## G_impute[3]  0.9997011
## G_impute[4]  0.9992379
## G_impute[5]  0.9993689
## G_impute[6]  0.9996437
## G_impute[7]  0.9994207
## G_impute[8]  0.9992872
## G_impute[9]  0.9995031
## G_impute[10] 0.9995187
## G_impute[11] 0.9993691
## G_impute[12] 1.0004765
## G_impute[13] 1.0006038
## G_impute[14] 1.0005580
## G_impute[15] 0.9993679
## G_impute[16] 0.9997290
## G_impute[17] 0.9990766
## G_impute[18] 0.9998225
## G_impute[19] 0.9993099
## G_impute[20] 0.9995981
## G_impute[21] 0.9990800
## G_impute[22] 0.9993021
## G_impute[23] 0.9993839
## G_impute[24] 0.9991474
## G_impute[25] 1.0000509
## G_impute[26] 0.9994053
## G_impute[27] 0.9995403
## G_impute[28] 0.9993227
## G_impute[29] 0.9997388
## G_impute[30] 0.9992544
```

```
## G_impute[31] 0.9997312
## G_impute[32] 0.9992507
## G_impute[33] 0.9992633
## alpha_G      0.9996517
## alpha_B      1.0000003
## betaG_M      0.9999858
## betaB_G      0.9997448
## betaB_M      0.9992944
## etasqG       0.9995734
## rhoG         0.9999018
## etasqB       0.9998765
## rhoB         0.9996559
```

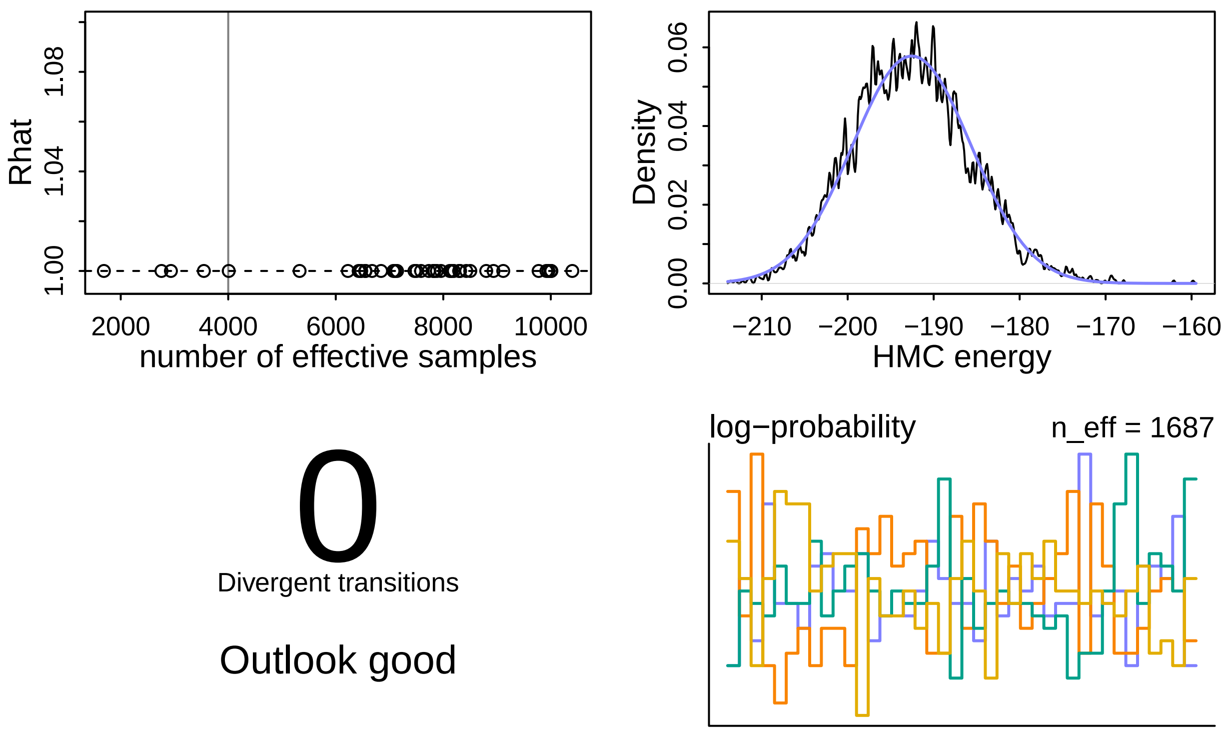
```r
dashboard(m2a)
```




```r
dashboard(m2b)
```

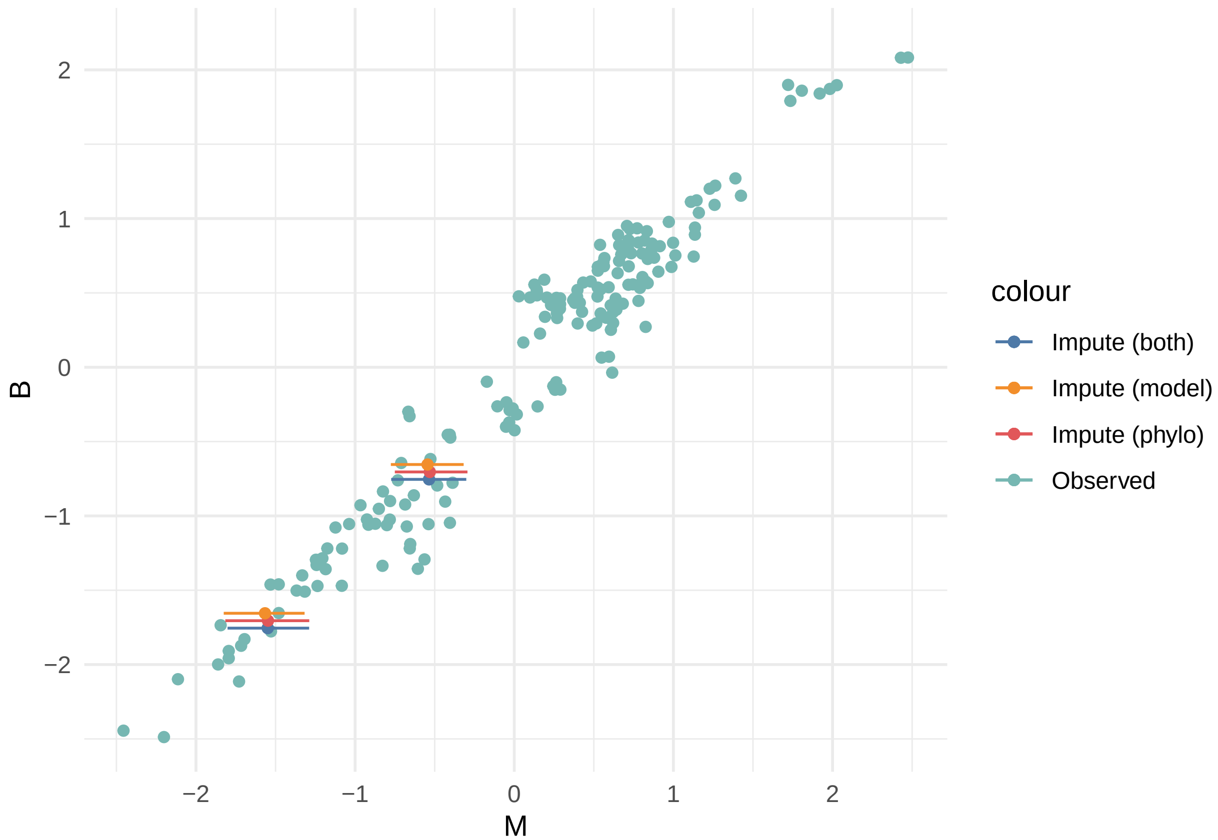



```r
dashboard(m2)
```



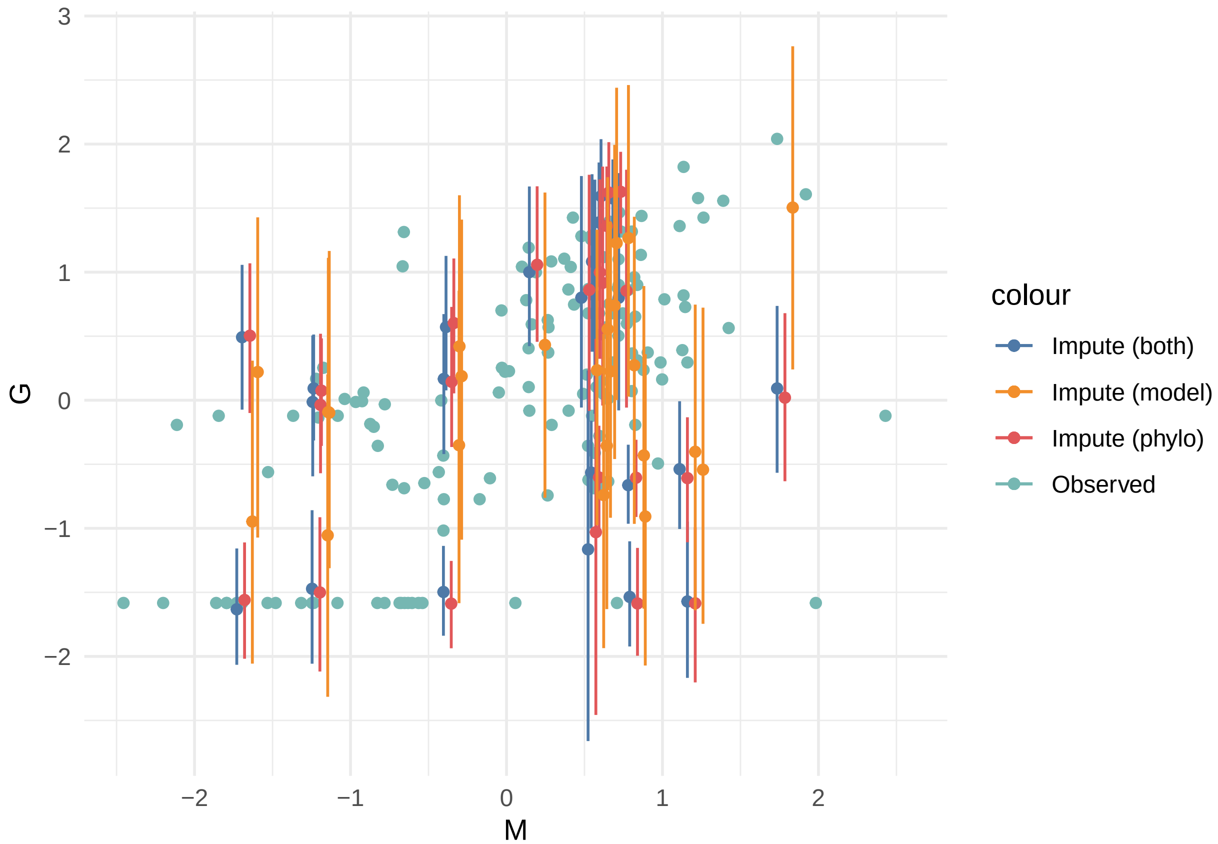

```r
p2a <- extract.samples(m2a)
p2b <- extract.samples(m2b)
p2  <- extract.samples(m2 )

df2a <- gen_df(p2a) %>%
```

```
    mutate(M = M+.05, B = B+.05)
df2b <- gen_df(p2b) %>%
    mutate(M = M+.10, B = B+.10)
df2  <- gen_df(p2 )

ggplot(df2) +
    geom_point(aes(x=M, y=B, colour='Observed'), na.rm=T) +
    geom_segment(aes(x=M_impute_lower, xend=M_impute_upper, y=B, yend=B, colour="Impute (bo
    geom_point(  aes(x=M_impute_median                    , y=B       , colour="Impute (bo
    geom_segment(aes(x=M_impute_lower, xend=M_impute_upper, y=B, yend=B, colour="Impute (ph
    geom_point(  aes(x=M_impute_median                    , y=B       , colour="Impute (ph
    geom_segment(aes(x=M_impute_lower, xend=M_impute_upper, y=B, yend=B, colour="Impute (mo
    geom_point(  aes(x=M_impute_median                    , y=B       , colour="Impute (mo
    scale_color_tableau() +
    theme_minimal()
```

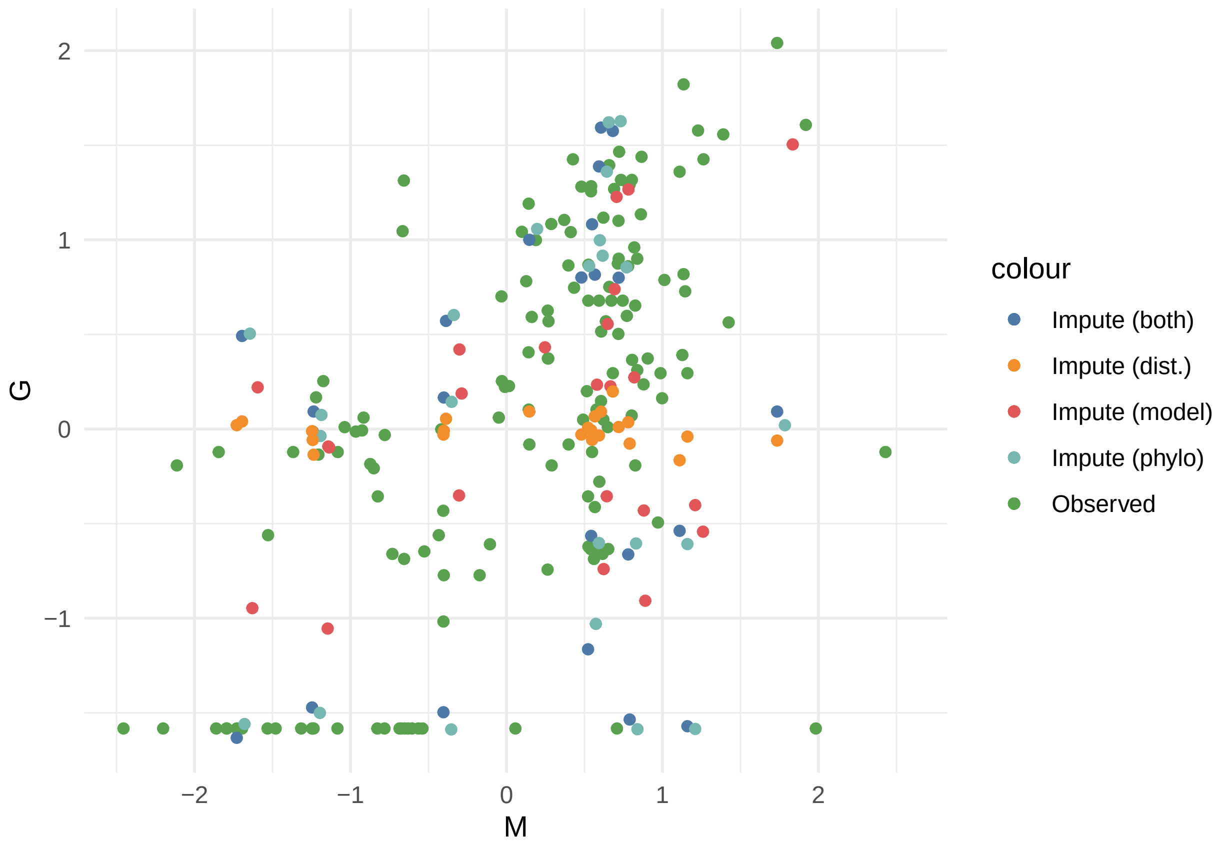



```
ggplot(df2) +
    geom_point(aes(x=M, y=G, colour='Observed'), na.rm=T) +
    geom_segment(aes(x=M, xend=M, y=G_impute_lower, yend=G_impute_upper, colour="Impute (bo
    geom_point(  aes(x=M        , y=G_impute_median                    , colour="Impute (bo
    geom_segment(aes(x=M, xend=M, y=G_impute_lower, yend=G_impute_upper, colour="Impute (ph
    geom_point(  aes(x=M        , y=G_impute_median                    , colour="Impute (ph
    geom_segment(aes(x=M, xend=M, y=G_impute_lower, yend=G_impute_upper, colour="Impute (mo
```

```
    geom_point(  aes(x=M         , y=G_impute_median                    , colour="Impute (mo
    scale_color_tableau() +
    theme_minimal()
```




```
ggplot(df2) +
    geom_point(aes(x=M, y=G, colour='Observed'), na.rm=T) +
    geom_point(  aes(x=M         , y=G_impute_median                    , colour="Impute (bo
    geom_point(  aes(x=M         , y=G_impute_median                    , colour="Impute (ph
    geom_point(  aes(x=M         , y=G_impute_median                    , colour="Impute (mo
    geom_point(  aes(x=M         , y=G_impute_median                    , colour="Impute (di
    scale_color_tableau() +
    theme_minimal()
```



Now we see that our imputation is now more correct, coming mainly from the phylogenetic information. Of course however the full model is the only academically honest model to show.

```
m3 <- cstan(file='../models/l18_BGM.stan', data=d, chains=4, cores=12, threads=3, iter=200
```

```
## Warning in readLines(stan_file): incomplete final line found on '../models/
## l18_BGM.stan'

## Running MCMC with 4 chains, at most 12 in parallel, with 3 thread(s) per chain...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)

## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 1 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat

## Chain 1 If this warning occurs sporadically, such as for highly constrained variable typ

## Chain 1 but if this warning occurs often then your model may be either severely ill-cond

## Chain 1

## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected b

## Chain 2 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat
```

```
## Chain 2 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 2 but if this warning occurs often then your model may be either severely ill-conc
## Chain 2
## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 3 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance mat
## Chain 3 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 3 but if this warning occurs often then your model may be either severely ill-conc
## Chain 3
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected b
## Chain 4 Exception: multi_normal_lpdf: LDLT_Factor of covariance parameter is not positiv
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable typ
## Chain 4 but if this warning occurs often then your model may be either severely ill-conc
## Chain 4
## Chain 3 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 4 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 3 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 4 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 3 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 3 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 4 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4 Iteration:  700 / 2000 [ 35%]  (Warmup)
```

```
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 3 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 4 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 3 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 4 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 3 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 4 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 4 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 3 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
```

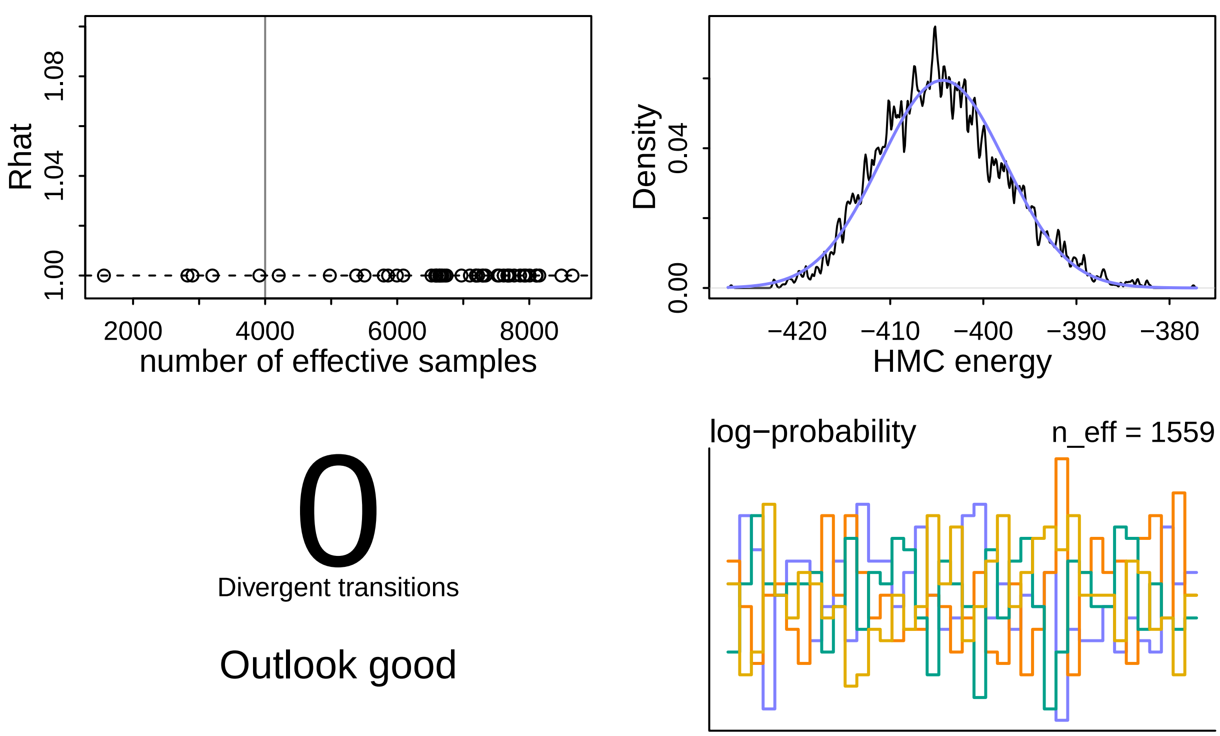
```
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 4 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3 finished in 275.4 seconds.
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4 finished in 283.2 seconds.
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 287.2 seconds.
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 287.9 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 283.4 seconds.
## Total execution time: 288.1 seconds.
```

```
precis(m3, depth=2)
```

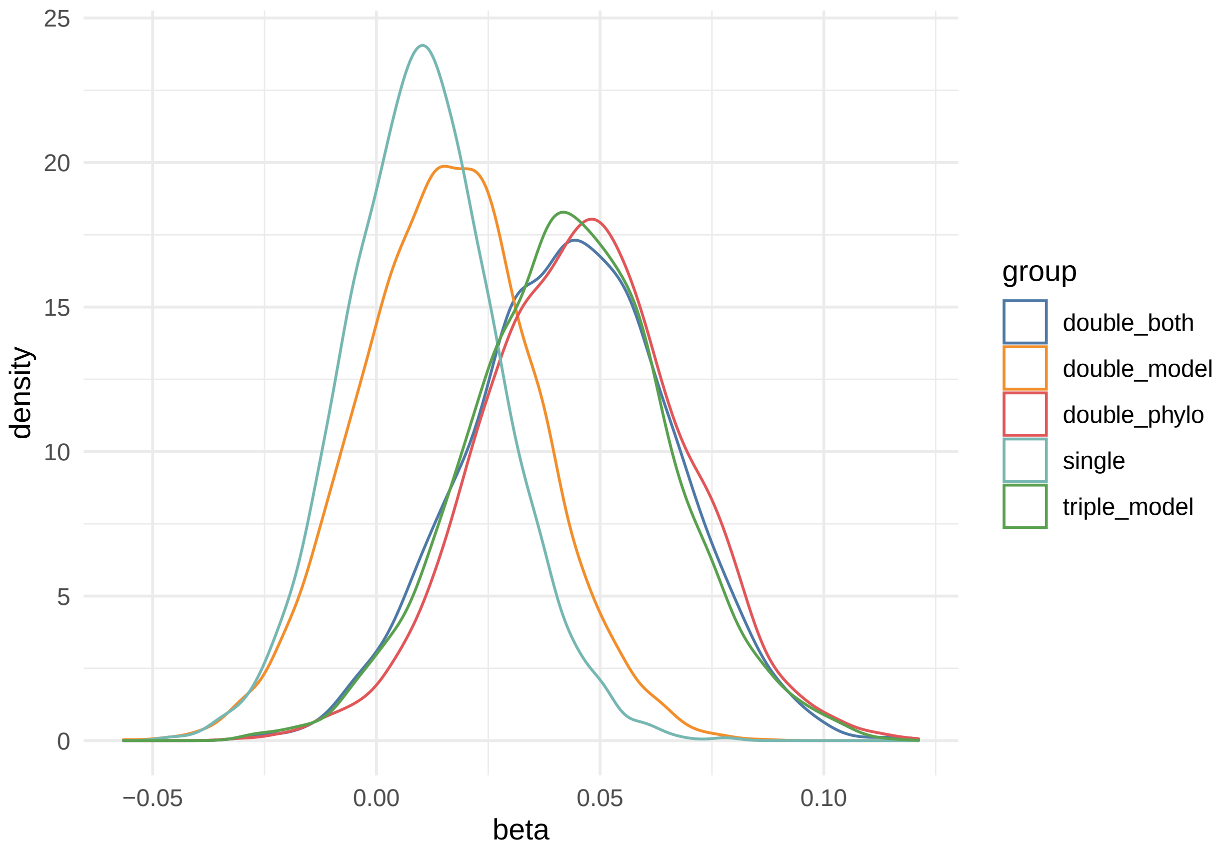
```
##                      mean          sd        5.5%       94.5%    n_eff     Rhat4
## M_impute[1]   -1.61002322 0.126329192 -1.81460440 -1.40603450 7612.247 0.9993198
## M_impute[2]   -0.55185852 0.111519055 -0.72933853 -0.37347123 7519.234 0.9993735
## G_impute[1]    0.16149325 0.339461322 -0.38615800  0.69530714 6747.694 1.0002003
## G_impute[2]   -0.66446815 0.189335951 -0.96747881 -0.36928028 7100.518 0.9995121
## G_impute[3]   -1.62832725 0.284283665 -2.08722055 -1.17311725 7193.901 0.9992926
## G_impute[4]    1.11789896 0.286485254  0.65721587  1.57968585 7861.873 0.9993346
## G_impute[5]    0.35662542 0.459026675 -0.35288670  1.09025620 7337.586 0.9994820
## G_impute[6]    0.91205392 0.248309247  0.51526666  1.31418235 8005.543 0.9994280
## G_impute[7]    0.38127259 0.193884153  0.07473594  0.69225363 7191.264 0.9994066
## G_impute[8]    0.59619018 0.278042239  0.15977322  1.04269880 6519.382 0.9995318
## G_impute[9]    0.65382811 0.525550221 -0.18111256  1.49940820 7546.355 0.9994860
## G_impute[10]   0.49181405 0.365193402 -0.09397317  1.07202740 5504.964 1.0001444
## G_impute[11]   0.56727020 0.329968681  0.03443949  1.08700595 7681.528 0.9990933
## G_impute[12]   0.83389787 0.579696933 -0.10135957  1.77799915 2823.423 1.0010590
## G_impute[13]   0.82126071 0.587357367 -0.10106513  1.73727855 2899.640 1.0018829
## G_impute[14]   0.81904500 0.570812711 -0.10589365  1.71802160 3200.783 1.0005342
## G_impute[15]   0.08916263 0.260055485 -0.32387094  0.49995076 6573.841 0.9991757
## G_impute[16]  -0.02721204 0.340333934 -0.55452134  0.51869877 6599.536 0.9995627
## G_impute[17]  -1.46540272 0.366374151 -2.05921850 -0.87266485 7772.295 0.9993050
## G_impute[18]   0.09127309 0.412055684 -0.57372615  0.74995311 8655.096 0.9992599
## G_impute[19]  -0.54062297 0.312861475 -1.05187935 -0.04829028 6685.219 0.9992198
## G_impute[20]  -0.57605568 0.265301883 -0.99209053 -0.15262199 6717.892 0.9994075
```

```
## G_impute[21] -1.53348038 0.272617875 -1.95857070 -1.10001725 7945.983 0.9994647
## G_impute[22]  1.06555326 0.450172353  0.34781813  1.76451515 7704.405 0.9993125
## G_impute[23]  1.00660191 0.371311816  0.42497294  1.60516005 8111.525 0.9994811
## G_impute[24] -1.50036575 0.218807821 -1.84922045 -1.15797475 5862.195 0.9995954
## G_impute[25] -1.57153482 0.395254690 -2.21720080 -0.94968966 7304.879 0.9995696
## G_impute[26]  1.58353339 0.206936065  1.24945095  1.91166255 6638.298 0.9996678
## G_impute[27]  1.59787569 0.278174422  1.14865560  2.03323770 7668.501 0.9995463
## G_impute[28]  1.38488972 0.290190629  0.92151148  1.84597630 6637.155 0.9997785
## G_impute[29] -1.17261155 0.933671330 -2.70410025  0.29163686 7288.893 1.0000344
## G_impute[30] -1.10257004 0.483636275 -1.87112795 -0.32200458 8151.857 0.9995774
## G_impute[31] -0.08066230 0.245238354 -0.47207545  0.30281348 6972.883 0.9995877
## G_impute[32] -0.17406101 0.365003407 -0.74513730  0.40228286 7919.891 0.9993696
## G_impute[33] -0.70930409 0.316782803 -1.21868760 -0.19612862 6593.341 0.9997729
## alpha_G      -0.26726920 0.771491150 -1.50444490  0.95754675 8104.485 0.9993696
## alpha_B      -0.05250516 0.839871288 -1.40820870  1.27082675 8486.977 0.9990943
## betaG_M       0.27125528 0.136613962  0.05097055  0.49298124 5994.701 0.9993768
## betaB_G       0.04300948 0.022090809  0.00756077  0.07798751 6664.437 0.9995603
## betaB_M       0.81750306 0.032217959  0.76442541  0.86767015 7225.497 0.9996625
## etasqM        2.93100038 0.258999416  2.52506370  3.34596760 4205.914 0.9999058
## rhoM          0.23939477 0.038016004  0.18374802  0.30590515 3913.613 0.9996365
## etasqG        2.89341311 0.249591065  2.49843685  3.28565265 5378.611 0.9995200
## rhoG          0.85020036 0.131973079  0.65289151  1.07850110 4979.744 1.0000857
## etasqB        2.96683424 0.252059342  2.56082800  3.36794165 6092.597 0.9992208
## rhoB          0.02043085 0.004356383  0.01426844  0.02782524 5795.941 0.9995262
```

```
dashboard(m3)
```



```
p3  <- extract.samples(m3)
df3 <- gen_df(p3)

GonB <- list( single=p1$betaB_G
            , double_both=p2$betaB_G
            , double_phylo=p2a$betaB_G
            , double_model=p2b$betaB_G
            , triple_model=p3$betaB_G )
df_b <- map_dfr(GonB, ~data.frame(beta=.), .id='group')


ggplot(df_b) +
    geom_density(aes(x=beta, colour=group)) +
    scale_color_tableau() +
    theme_minimal()
```



It is clear from this that while a lot of the more developed model produces the same result by consuming more of the available information the simple models do no capture the correct inference as precisely.

Censored Observation

Blending togethor missing data and measurement error is censored observation. For instance time that academics stay in academia post Ph.D. We do not know for sure when current academics will leave, but we can say something about the distribution of people remaining, and we already have information about a minimum value for these people. It is a mistake to ignore events that might yet happen.

Lecture 19: GLM Madness

We can use GLM and GLMM to get effective fits for very unreasonable amount of problems. However a scientifically motivated model is always going to contribute more to our understanding of problems. Vitally this include misfitting models!!! Where the fit fails informs the weaknesses in the scientific model.

To use a scientific model remember the constraints on your data, you can often simplify your model. For instance take modelling weight and height

$$W = k\pi p^2 H^3$$

can be simplified to

$$W = H^3$$

by normalising your height and weight to have 1 be the average for both.

State Based (Conditional) Probability

Simply code it up in a for loop in STAN. This allows you to attempt to decompose states for instance

$$\begin{aligned}
Y_i &\sim \text{Categorical}(\theta_i) \\
\theta_i &= \sum_S p_S \Pr(Y = i|S) \\
p_j &\sim \text{Dirchlet}(\vec{4})
\end{aligned}$$

```
data(Boxes_model)
cat(Boxes_model)
```

```
##
## data{
##     int N;
##     int y[N];
##     int majority_first[N];
```

```
## }
## parameters{
##     simplex[5] p;
## }
## model{
##     vector[5] phi;
##
##     // prior
##     p ~ dirichlet( rep_vector(4,5) );
##
##     // probability of data
##     for ( i in 1:N ) {
##         if ( y[i]==2 ) phi[1]=1; else phi[1]=0; // majority
##         if ( y[i]==3 ) phi[2]=1; else phi[2]=0; // minority
##         if ( y[i]==1 ) phi[3]=1; else phi[3]=0; // maverick
##         phi[4]=1.0/3.0;                          // random
##         if ( majority_first[i]==1 )              // follow first
##             if ( y[i]==2 ) phi[5]=1; else phi[5]=0;
##         else
##             if ( y[i]==3 ) phi[5]=1; else phi[5]=0;
##
##         // compute log( p_s * Pr(y_i|s )
##         for ( j in 1:5 ) phi[j] = log(p[j]) + log(phi[j]);
##         // compute average log-probability of y_i
##         target += log_sum_exp( phi );
##     }
## }
```

We refer to the choices as emissions and we want to extract the strategies; the latent states. A decoding is often very noisy, but it really is the only honest way to report the results.

Population Dynamics

For instance a predator–prey population dynamics. Scientifically we measure Lynxes L and Hares H. We can think of the change in population as a set of coupled differential equations

$$\frac{\partial N_H}{\partial t} = N_H \left(b_H - m_H N_L\right)$$
$$\frac{\partial N_L}{\partial t} = N_L \left(b_L N_H - m_L\right)$$

So we could model this taking into account our model has measurement error as we only have proxies of number of animals (trapping)

```r
data(Lynx_Hare_model)
cat(Lynx_Hare_model)
```

```
## functions {
##   real[] dpop_dt( real t,                 // time
##                  real[] pop_init,          // initial state {lynx, hares}
##                  real[] theta,             // parameters
##                  real[] x_r, int[] x_i) {  // unused
##     real L = pop_init[1];
##     real H = pop_init[2];
##     real bh = theta[1];
##     real mh = theta[2];
##     real ml = theta[3];
##     real bl = theta[4];
##     // differential equations
##     real dH_dt = (bh - mh * L) * H;
##     real dL_dt = (bl * H - ml) * L;
##     return { dL_dt , dH_dt };
##   }
## }
## data {
##   int<lower=0> N;              // number of measurement times
##   real<lower=0> pelts[N,2];    // measured populations
## }
## transformed data{
##   real times_measured[N-1];    // N-1 because first time is initial state
##   for ( i in 2:N ) times_measured[i-1] = i;
## }
## parameters {
##   real<lower=0> theta[4];      // { bh, mh, ml, bl }
##   real<lower=0> pop_init[2];   // initial population state
##   real<lower=0> sigma[2];      // measurement errors
##   real<lower=0,upper=1> p[2];  // trap rate
## }
## transformed parameters {
##   real pop[N, 2];
##   pop[1,1] = pop_init[1];
##   pop[1,2] = pop_init[2];
##   pop[2:N,1:2] = integrate_ode_rk45(
##     dpop_dt, pop_init, 0, times_measured, theta,
##     rep_array(0.0, 0), rep_array(0, 0),
##     1e-5, 1e-3, 5e2);
## }
## model {
##   // priors
##   theta[{1,3}] ~ normal( 1 , 0.5 );    // bh,ml
##   theta[{2,4}] ~ normal( 0.05, 0.05 ); // mh,bl
##   sigma ~ exponential( 1 );
##   pop_init ~ lognormal( log(10) , 1 );
```

```
##   p ~ beta(40,200);
##   // observation model
##   // connect latent population state to observed pelts
##   for ( t in 1:N )
##     for ( k in 1:2 )
##       pelts[t,k] ~ lognormal( log(pop[t,k]*p[k]) , sigma[k] );
## }
## generated quantities {
##   real pelts_pred[N,2];
##   for ( t in 1:N )
##     for ( k in 1:2 )
##       pelts_pred[t,k] = lognormal_rng( log(pop[t,k]*p[k]) , sigma[k] );
## }
```

Note we use the familiar `integrate_ode_rk45`. This is just a brief presentation of other types of analysis possible as well. The point is that if every field is just using linear regression and t–tests then they are just throwing out linear regression. For this we should postpone the statistics, and start with scientific reasoning. While scientific models are flawed, despite their flaws they are productive.

Lecture 20: Horoscopes

The general reporting template

Taking the dyad model we want to provide the DAG

$$\begin{tikzcd}
 & TBA \arrow[d] & \\
HA \arrow[ur] \arrow[r] \arrow[dr] & GAB & HB \arrow[ul] \arrow[l] \arrow[dl] \\
 & TAB \arrow[u] &
\end{tikzcd}$$

as well as the model

$$
\begin{aligned}
G_{AB} &\sim \text{Poisson}(\lambda_{AB}) \\
\log(\lambda_{AB}) &= \alpha + T_{AB} + G_A + R_B \\
G_{BA} &\sim \text{Poisson}(\lambda_{BA}) \\
\log(\lambda_{BA}) &= \alpha + T_{BA} + G_B + R_A \\
\begin{pmatrix} T_{AB} \\ T_{BA} \end{pmatrix} &\sim \text{MVNormal}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma^2 & \rho\sigma^2 \\ \rho\sigma^2 & \sigma^2 \end{bmatrix} \right) \\
\begin{pmatrix} G_A \\ R_A \end{pmatrix} &\sim \text{MVNormal}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, R_{GR}, S_{GR} \right) \\
\alpha &\sim \text{Normal}(0, 1) \\
\sigma, S_{GR} &\sim \text{Exponential}(1) \\
\rho, R_{GR} &\sim \text{LJKCorr}(2).
\end{aligned}
$$

We also want to provide description of the method to, here is **a** template for a minimal honest best practice:

Model Description

To estimate reciprocity within dyads, we model the correlation within dyads in giving, using a multilevel mixed–membership model (textbook citation).

- From reviewers, there is a classical reviewer who thinks that if the science were done well, it would have simple stats or no stats at all (*Good science doesn't need complex stats*). **This is ridiculous**
  - Rebut it to the editor, and switch to the causal model not the statistics. Justify the stratification causally, which requires your statistical complexity.
  - Just because a simple procedure qualitatively gives the same answer doesn't mean we should use more statistically rigorous methods. The more complicated one typically look for confounds and unit heterogeneity, and we don't want to be right because we got lucky. Knowledge is justified true belief, not just belief.
  - The review comment is better in general, change discussion from statistics to causal models, here justification is king. Be sure to remain civil.

Do Calculus

To control for confounding from generalised giving and receiving, as indicated by (DAG link), we stratify by giving and receiving by household. The full model with priors is (equ link).

- Why we think the model works
- **Priors were chosen through prior predictive simulation so that pre–data predictions span the range of scientifically plausible outcomes. In the results, we explicitly compare posterior prediction and prior, so that the impact of the sample is obvious.**

Simulation

We estimate the posterior distribution using Hamiltonian Monte Carlo as implemented in STAN (version, citation).

- To further science we want good software, to get good software they need measures of success to be able to keep doing what they are doing; such as citations, so cite all software.

Diagnostic

We validated the model on simulated data and assessed convergence by inspection of trace plots, R–hat values, and effective sample sizes. Diagnostics are reported in (Appendix Link) and all results can be replicated using the code available at (link).

Results

- Avoid all too easy misunderstandings, avoid language that would confuse non–Bayesian familiar scientists, show posteriors and avoid things easily confused for confidence intervals and cite introductionary review papers in the field that can help people
- Talk about any missing values
- Describe control variable interpretation, make sure it is obvious some cannot be interpreted
- Densities are better than intervals, sample realisations (functions)

Session

```r
sessionInfo()
```

```
## R version 4.2.0 (2022-04-22)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.4 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0
##
## locale:
##  [1] LC_CTYPE=C.UTF-8       LC_NUMERIC=C           LC_TIME=C.UTF-8
##  [4] LC_COLLATE=C.UTF-8     LC_MONETARY=C.UTF-8    LC_MESSAGES=C.UTF-8
##  [7] LC_PAPER=C.UTF-8       LC_NAME=C              LC_ADDRESS=C
## [10] LC_TELEPHONE=C         LC_MEASUREMENT=C.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
```

```
##  [1] digest_0.6.29       forcats_0.5.1       stringr_1.4.0
##  [4] dplyr_1.0.9         purrr_0.3.4         readr_2.1.2
##  [7] tidyr_1.2.0         tibble_3.1.7        tidyverse_1.3.1
## [10] ape_5.6-2           gtools_3.9.2.1      rethinking_2.21
## [13] cmdstanr_0.5.2      rstan_2.26.11       StanHeaders_2.26.11
## [16] ggthemes_4.2.4      ggplot2_3.3.6       rmarkdown_2.14
## [19] knitr_1.39
##
## loaded via a namespace (and not attached):
##  [1] nlme_3.1-157         fs_1.5.2             matrixStats_0.62.0
##  [4] lubridate_1.8.0      httr_1.4.3           tensorA_0.36.2
##  [7] tools_4.2.0          backports_1.4.1      utf8_1.2.2
## [10] R6_2.5.1             DBI_1.1.2            colorspace_2.0-3
## [13] withr_2.5.0          tidyselect_1.1.2     gridExtra_2.3
## [16] prettyunits_1.1.1    processx_3.5.3       curl_4.3.2
## [19] compiler_4.2.0       rvest_1.0.2          cli_3.3.0
## [22] xml2_1.3.3           labeling_0.4.2       bookdown_0.26
## [25] posterior_1.2.1      scales_1.2.0         checkmate_2.1.0
## [28] mvtnorm_1.1-3        callr_3.7.0          pkgconfig_2.0.3
## [31] htmltools_0.5.2      highr_0.9            dbplyr_2.1.1
## [34] fastmap_1.1.0        rlang_1.0.2          readxl_1.4.0
## [37] rstudioapi_0.13      shape_1.4.6          farver_2.1.0
## [40] generics_0.1.2       jsonlite_1.8.0       distributional_0.3.0
## [43] inline_0.3.19        magrittr_2.0.3       loo_2.5.1
## [46] Rcpp_1.0.8.3         munsell_0.5.0        fansi_1.0.3
## [49] abind_1.4-5          lifecycle_1.0.1      stringi_1.7.6
## [52] yaml_2.3.5           MASS_7.3-57          pkgbuild_1.3.1
## [55] grid_4.2.0           crayon_1.5.1         lattice_0.20-45
## [58] haven_2.5.0          hms_1.1.1            ps_1.7.0
## [61] pillar_1.7.0         codetools_0.2-18     stats4_4.2.0
## [64] reprex_2.0.1         glue_1.6.2           evaluate_0.15
## [67] V8_4.2.0             data.table_1.14.2    RcppParallel_5.1.5
## [70] modelr_0.1.8         vctrs_0.4.1          tzdb_0.3.0
## [73] cellranger_1.1.0     gtable_0.3.0         assertthat_0.2.1
## [76] xfun_0.31            broom_0.8.0          coda_0.19-4
## [79] ellipsis_0.3.2
```

References

Cinelli, Carlos, Andrew Forney, and Judea Pearl. n.d. "A Crash Course in Good and Bad Controls." https://ssrn.com/abstract=3689437.

Kurz, A. Solomon. 2021. *Statistical Rethinking with Brms, Ggplot2, and the Tidyverse: Second Edition*. Version 0.2.0. https://bookdown.org/content/4857/.

McElreath, Richard. 2020. *rethinking R Package*. https://xcelab.net/rm/software/.

———. 2022. *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. https://xcelab.net/rm/statistical-rethinking/.

Visible Earth. 2006. "Blue Marble." NASA.

Westreich, Daniel, and Sander Greenland. 2013. "The Table 2 Fallacy: Presenting and Interpreting Confounder and Modifier Coefficients." *American Journal of Epidemiology* 177 (4): 292–98. https://doi.org/10.1093/aje/kws412.