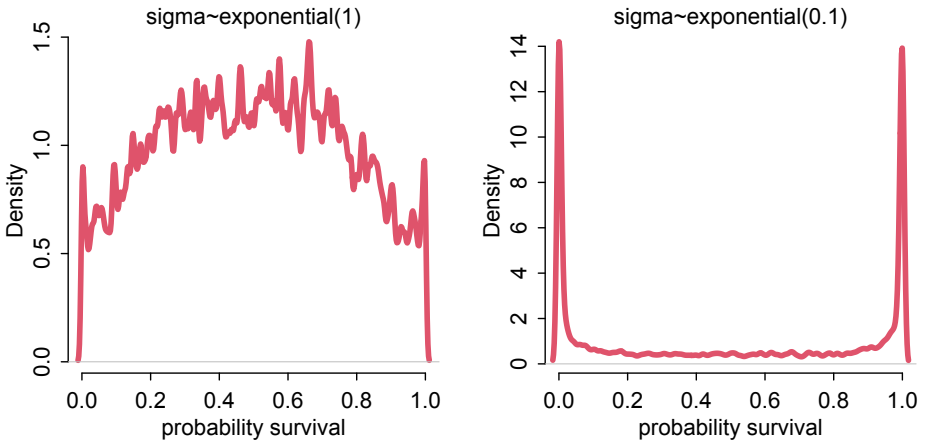
STATISTICAL RETHINKING 2024
WEEK 6 SOLUTIONS

1. Simulating varying effect priors is in principle like simulating any other priors. The only difference is that the parameters have an implied order now, because some parameters depend upon others. So in this problem we must simulate σ and $\bar{\alpha}$ first, and then we can simulate the individual tank α_T variables. Here is how I did it:

```
n <- 1e4
sigma <- rexp(n,1)
abar <- rnorm(n,0,1)
aT <- rnorm(n,abar,sigma)
dens(inv_logit(aT),xlim=c(0,1),adj=0.1,lwd=4,col=2)
```

This simulates 1e4 tanks from the prior. Then it plots the implied survival distribution on the probability scale. Before showing that plot, I'll also simulate from a prior in which $\sigma \sim \text{Exponential}(0.1)$. Showing both:



Increasing the variantion across tanks, by making the σ distribution wider, pushes prior survival up against the floor and ceiling of the outcome space. This is the same phenomenon you saw before for ordinary logit models. The key lesson again is that flat priors on one scale are not necessarily flat on another.

2. I will use the suggested matrix of parameters approach.

```
library(rethinking)
data(reedfrogs)
d <- reedfrogs
```

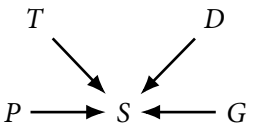
```
dat <- list(
    S = d$surv,
    D = d$density,
    T = 1:nrow(d),
    P = ifelse( d$pred=="no" , 1L , 2L ),
    G = ifelse( d$size=="small" , 1L , 2L ) )

m2 <- ulam(
    alist(
        S ~ binomial( D , p ),
        logit(p) <- a[T] + b[P,G],
        a[T] ~ normal( 0 , sigma ),
        matrix[P,G]:b ~ normal( 0 , 1 ),
        sigma ~ exponential( 1 )
    ), data=dat , chains=4 , cores=4 , log_lik=TRUE )
precis(m2,3,pars=c("b","sigma"))
```

```
        mean   sd  5.5% 94.5% n_eff Rhat4
b[1,1]  2.36 0.29  1.89  2.84  1108  1.01
b[1,2]  2.50 0.31  2.00  2.98  1282  1.00
b[2,1]  0.43 0.25  0.06  0.81   673  1.00
b[2,2] -0.43 0.25 -0.81 -0.03   720  1.00
sigma   0.74 0.15  0.51  0.98   455  1.01
```

The parameters are in order from top to bottom: no-pred/small, no-pred/large, pred/small, pred/large. The curious thing is not that survival is lower with predation, but rather that it is lowest for large tadpoles, `b[2,2]`. This is a strong interaction that would be missed if we had made the effects purely additive with one another (on the log-odds scale). The Vonesh & Bolker paper that these data come from goes into this interaction in great depth.

The problem asked for a justification of the model in terms of the DAG. Here's the DAG:



This is an experiment, so we know the treatments P, G, and D are not confounded. At least not in any obvious way. And then unobserved tank effects T also moderate the influence of the treatments. The model I used tries to estimate how P and G moderate one another. It ignores D, which we are allowed to do, because it is not a confound, just a competing cause. But I include tanks, which is also just a competing cause. Including competing causes helps with precision, if nothing else.

What the DAG does not justify is the full interaction matrix that I used. DAGs contain no information about how variables interact or not to produce outcomes.

They just show inputs and outputs. To justify any particular statistical model, you need more than the DAG.

3. We can copy the basic treatment model from lecture to begin:

```
library(rethinking)
data(Trolley)
d <- Trolley
dat <- list(
    R = d$response,
    A = d$action,
    I = d$intention,
    C = d$contact )

# model without individuals, with treatments
mRX <- ulam(
    alist(
        R ~ dordlogit(phi,alpha),
        phi <- bA*A + bI*I + bC*C,
        c(bA,bI,bC) ~ normal(0,0.5),
        alpha ~ normal(0,1)
    ) , data=dat , chains=4 , cores=4 )
```

This is just a plain ordered logit model with three indicator variables in the linear model `phi`. For the coefficients, we get:

```
    mean   sd  5.5% 94.5% n_eff Rhat4
bC -0.94 0.05 -1.03 -0.86  1771     1
bI -0.71 0.04 -0.77 -0.65  2034     1
bA -0.70 0.04 -0.76 -0.63  1565     1
```

These are just what you've seen before, each reduces the response, but contact (C) is the biggest effect.

Now to add individual varying intercepts, we need an index variable and then a vector of parameters, one for each individual. The `id` variable in the original data is a string factor, not a true index variable, so we need to coerce it to integer. There's also a utility function in `rethinking` called `check_index()` that I'll use to see if this works:

```
dat$id <- as.integer(d$id)
check_index(dat$id)
```

```
Length: 331
Range: 1 / 331
```

Looking good: 331 individuals, and the index values start at 1 and are continuous.

There are two tricky issues in adding individual varying effects to this model. The first is whether to use a centered or non-centered parameterization of the multilevel prior. The second is whether or not to have a parameter for the average individual.

For the first issue, a non-centered prior will work much better in this case, because there are many individuals and there isn't much data from each, only 30 responses per individual. So the prior will do a lot of work in partial pooling. You can try the centered version, and you'll find it samples very poorly.

For the second issue, a parameter for the average individual isn't strictly necessary, because the cutpoint parameters already encode the average individual. We just need offsets for each individual. But you can include another parameter for the average individual. You'll get the same treatment estimates. Ironically, in this case the model samples better when you over-parameterize it and include the individual mean. So I'll do that.

This is my suggested code. This model can take a while (10-40 minutes, depending upon your hardware) to sample. So be patient. I suggest starting with a small number of samples, like `iter=500`. Then you can get a sense whether the model is sampling well and do a longer run when your computer has time. But often 250 non-warmup samples from each of 4 chains = 1000 samples is plenty. I use 1000 in the code below. So change the `iter` argument before running it yourself.

```
# model with individual varying intercepts, non-centered
mRXid_nc <- ulam(
    alist(
        R ~ dordlogit(phi,alpha),
        phi <- bA*A + bI*I + bC*C + k[id],
        transpars> vector[331]:k <<- kbar + z*tau,
        # old priors
        c(bA,bI,bC) ~ normal(0,0.5),
        alpha ~ normal(0,1),
        # new priors
        z[id] ~ normal(0,1),
        kbar ~ normal(0,0.5),
        tau ~ exponential(1)
    ) , data=dat , chains=4 , cores=4 , iter=1000 )
precis(mRXid_nc)
```

```
      mean   sd  5.5% 94.5% n_eff Rhat4
bC   -1.25 0.05 -1.34 -1.17  3129  1.00
bI   -0.94 0.04 -1.00 -0.89  3906  1.00
bA   -0.94 0.04 -1.01 -0.87  2997  1.00
kbar  0.79 0.33  0.28  1.31  1665  1.00
tau   1.88 0.08  1.76  2.00   350  1.01
```

The scale `tau` is not sampling as well as the other parameters. But it gets the job done.

All of the treatment coefficients are even more negative now, but the rank order hasn't changed. The individual variation in scale use was masking the size of the treatment effects, it seems.

If individual variation had such a large impact, there is another competing cause that might also be worth considering: the story. There are 12 different "stories" in

the experiment, each a different kind of scenario. We could include these as varying intercepts as well. Here's a model:

```
dat$S <- as.integer(d$story)
check_index(dat$S)
mRXidS <- ulam(
    alist(
        R ~ dordlogit(phi,alpha),
        phi <- bA*A + bI*I + bC*C + k[id] + s[S],
        transpars> vector[331]:k <<- kbar + z*tau,
        # old priors
        c(bA,bI,bC) ~ normal(0,0.5),
        alpha ~ normal(0,1),
        # new priors
        s[S] ~ normal(0,sigma),
        z[id] ~ normal(0,1),
        kbar ~ normal(0,0.5),
        tau ~ exponential(1),
        sigma ~ exponential(1)
    ) , data=dat , chains=4 , cores=4 , iter=500 )
precis(mRXidS)
```

```
       mean   sd  5.5% 94.5% n_eff Rhat4
bC    -1.80 0.08 -1.93 -1.68  1008  1.00
bI    -0.91 0.05 -0.99 -0.84  1298  1.00
bA    -1.20 0.06 -1.29 -1.11   904  1.00
kbar   0.86 0.34  0.32  1.41   377  1.01
tau    1.94 0.08  1.80  2.07    94  1.01
sigma  0.62 0.16  0.43  0.89   361  1.01
```

I should run that longer, but notice the change in the treatment estimates. These competing causes are not confounds, but in non-linear models they can mislead us just as much. To see why the treatment estimates changed, let's look at the story effects:

```
precis(mRXidS,depth=2,pars="s")
```

```
       mean   sd  5.5% 94.5% n_eff Rhat4
s[1]   1.16 0.20  0.88  1.47   121  1.04
s[2]  -0.18 0.18 -0.45  0.11   126  1.04
s[3]   0.04 0.18 -0.22  0.35   113  1.05
s[4]   0.66 0.19  0.39  0.98   119  1.05
s[5]  -0.09 0.18 -0.35  0.21    86  1.06
s[6]   0.07 0.19 -0.20  0.39   129  1.05
s[7]  -0.50 0.19 -0.78 -0.19   123  1.04
s[8]   0.27 0.19 -0.02  0.59   133  1.04
s[9]  -0.42 0.19 -0.70 -0.11   116  1.04
s[10]  0.94 0.19  0.66  1.25   115  1.04
```

```
s[11] -0.38 0.19 -0.65 -0.07   122  1.05
s[12] -0.27 0.18 -0.54  0.03    91  1.05
```

And those chains definitely need to be run longer (or made non-centered). But you can see that there are couple of stories, numbers 1 and 10, with much higher offsets than the others. This could matter, because not all stories can be combined with all treatments. Number 1 never implies intent, for example, and number 10 always does:

```
table(dat$S,dat$I)
```

```
      0   1
 1  662   0
 2  662   0
 3  331 993
 4  662 662
 5  662   0
 6  331 331
 7    0 662
 8  662   0
 9  331 331
 10   0 662
 11 331 662
 12 662 331
```

Some people would say that treatment is confounded by story in this situation. Regardless of what you call it, including story effects makes a big difference.