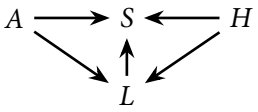
STATISTICAL RETHINKING 2023
WEEK 9 SOLUTIONS

1. Before we get started on modeling a function to relate age to hunting success, let's think about this problem causally. Age could work through multiple causal paths. We want to estimate all of them. From what I've given you, the causes to consider are age (A), trip duration (L), and unmeasured features of each hunter (H). A minimalist view of how these causes interact to influence success (S) might be:



So the total causal effect of age acts partly through any influence it has on the duration of each trip, if in fact duration has much influence on success. So we do not want to strategy by duration (L). We probably should stratify by hunter, but we'll do that in a later problem.

The problem now is find some reasonable function to describe the causal effect of age on success. Remember: Getting the function right is as important as getting the arrows right. I am going to work my way up to a satisfactory age function. A linear model is obviously not a good starting place, because any age-related increases in success must taper off at some point—80 year olds can't be getting more skilled at the same rate they did when they were 20. So I'll try log(age), which imposes diminishing returns in an automatic way. Here's the model, just a bernoulli model:

```
library(rethinking)
data(Achehunting)
d <- Achehunting
dat <- list(
    S = ifelse(d$kg.meat>0,1,0),
    A = standardize(log(d$age)) )

# log age model
m1a <- ulam(
    alist(
        S ~ bernoulli(p),
        logit(p) <- a + bA*A,
        a ~ normal(0,1),
        bA ~ normal(0,0.5)
    ) , data=dat , chains=4 , cores=4 )
precis(m1a)
```

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

```
a  -0.09 0.02 -0.12 -0.06  1577     1
bA  0.12 0.02  0.09  0.14   995     1
```

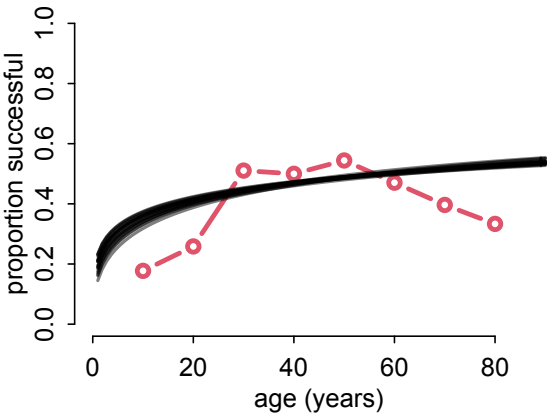
The precis output is not so useful, but you can check the diagnostics and make a quick assessment that the coefficient on log(age) is positive, indicating success increases with age.

Let's plot the posterior function though. It can be hard to plot posterior predictives for Bernoulli data, because all the observations are at 0 or 1. So instead let's bin the age values and compute proportions of success in each bin. This will do it:

```
# round age to nearest decade
Ar <- round( d$age / 10 ) * 10
Aseq <- c(10,20,30,40,50,60,70,80)
SA <- sapply( Aseq , function(a) mean(dat$S[Ar==a]) )
plot( Aseq , SA , ylim=c(0,1) , xlim=c(0,90) , type="b" , lwd=3 ,
    col=2 , xlab="age (years)" , ylab="proportion successful" )
```

And now to overlay the posterior function (20 draws from posterior). This code is only awkward because the plot has years on the horizontal, but the model was fit on standardized log years. So I need to do that adjustment instead the plotting code below.

```
post <- extract.samples(m1a)
for ( i in 1:20 )
    curve( inv_logit(
        post$a[i] + post$bA[i]*( log(x)-3.79 )/0.335 ) ,
        add=TRUE , lwd=2 , col=grau() , from=1 )
```



This is surely one of the worst fits I have ever produced. What is needed is something that isn't monotonic, something that can rise and then fall. A simply polynomial would work. But you know I am not a fan of polynomials. So I'm going to revive one of those growth models that I used in a previous homework, when I modeled

dinosaur growth. Let the probability of success at an age A be:

$$p(A) = \alpha(1 - \exp(-\beta_1 A))\exp(-\beta_2 A)$$

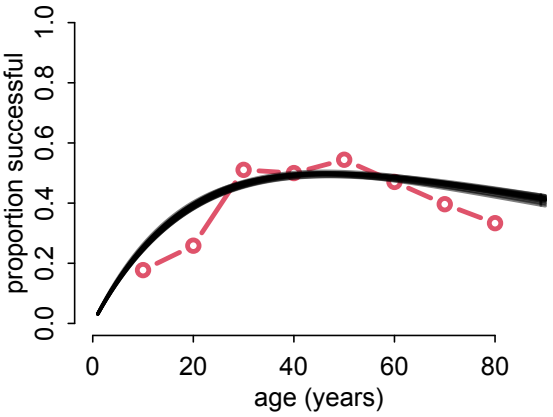
The parameter α sets the maximum, so we constrain it to be between zero and one. The β parameters control the rates of increase (β_1) and decrease (β_2).

Now it's just a simple matter of sticking this function into the model. We also need to add age on the natural scale to the data. I'll normalize it though by dividing by a reference age of 80. This just makes the fitting easier, because the coefficients won't multiply such large values.

```
dat$A2 <- d$age / 80
m1b <- ulam(
    alist(
        S ~ bernoulli(p),
        p <- a*(1-exp(-b1*A2))*exp(-b2*A2),
        a ~ beta(4,4),
        c(b1,b2) ~ exponential(2)
    ) , data=dat , chains=4 , cores=4 )
```

Plotting:

```
post <- extract.samples(m1b)
for ( i in 1:20 ) with( post ,
    curve( a[i]*(1-exp(-b1[i]*x/80))*exp(-b2[i]*x/80) ,
    add=TRUE , lwd=2 , col=grau() , from=1 ) )
```



Much better. Still looks like it rises too quickly at the start. Do babies really learn to hunt so quickly? We could fix this by shifting the age where the functions goes to zero. Or we could impose some increasing returns on the increasing portion of the function. Let's try that. Add one more parameter:

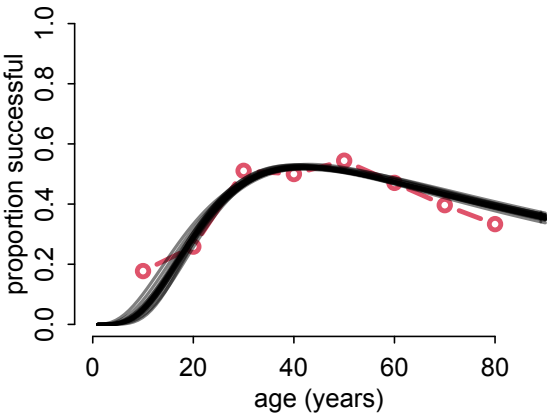
$$p(A) = \alpha(1 - \exp(-\beta_1 A))^\gamma \exp(-\beta_2 A)$$

I've inserted a γ, which manages the "elasticity" of the term that increases with age. If $\gamma > 1$, then the curve should accelerate at first. Let's try fitting it:

```
m1c <- ulam(
    alist(
        S ~ bernoulli(p),
        p <- a*exp(-b2*A2)*(1-exp(-b1*A2))^g,
        a ~ beta(4,4),
        g ~ exponential(0.5),
        c(b1,b2) ~ exponential(2)
    ) , data=dat , chains=4 , cores=4 )
```

Plotting:

```
post <- extract.samples(m1c)
for ( i in 1:20 ) with( post ,
    curve( a[i]*exp(-b2[i]*x/80)*(1-exp(-b1[i]*x/80))^g[i] ,
    add=TRUE , lwd=2 , col=grau() , from=1 ) )
```



That is much more satisfying. This has been a curve-fitting exercise. So it can be interesting to compare the PSIS scores of these models. To do so, run each model again after adding `log_lik=TRUE` to the `ulam` call. I do not do that by default, because there are 14-thousand observations here and log_lik produces a full posterior for each observation. That's a lot of output. But go ahead and do it and be patient with your computer. Then:

```
compare( m1a , m1b , m1c , func=PSIS )
```

```
       PSIS    SE dPSIS   dSE pPSIS weight
m1c 19590.7 32.02   0.0    NA   3.0      1
m1b 19704.7 16.96 114.0 18.09   1.8      0
m1a 19838.6 17.20 247.8 27.15   1.9      0
```

This echos what you can see in the posterior prediction plots, that the third model does much better.

And the story it tells is that success increases rapidly during the teens and then declines slowly after middle age. Note also that the success rate never rises much above 50% at its peak. Hunting is hard.

It's reasonable now to ask whether we've overfit, because we chose the function by iteratively looking at model fits. It's possible. The function is defensible in terms of its biological features. But if we had a different sample, maybe we would have chosen to model other biological features. Research is iterative. Just be transparent about it.

2. I will let the two rate parameters, β_1 and β_2, vary by individual hunter. So we need two varying effects. I'll use a non-centered parameterization. But I'll also add some code to convert them to centered effects, to keep the linear model simple. The trick to adding varying effects to this model is to maintain the constraints on β_1 and β_2. They both have to be positive. So when we draw varying effects from a normal distribution, we have to then transform them to be positive before using them in the linear model. This means in practice that we set the varying effects on the log scale and then exp them before use.

Here's the code. This model might take a little while to run. It took about 20 minutes on my laptop. Start it up and then go for a walk.

```
# hunter index variable
dat$H <- as.integer(as.factor(d$id))
dat$NH <- max(dat$H)

m2 <- ulam(
    alist(
        S ~ bernoulli(p),
        p <- a*exp(-b2H[H]*A2)*(1-exp(-b1H[H]*A2))^g,
        # centered varying effects
        transpars> vector[NH]:b1H <<- exp(b1+V[1:NH,1]),
        transpars> vector[NH]:b2H <<- exp(b2+V[1:NH,2]),
        # non-centered varying effects
        transpars> matrix[NH,2]:V <-
            compose_noncentered( sigma_H , L_Rho_H , Z ),
        matrix[2,NH]:Z ~ normal( 0 , 1 ),
        cholesky_factor_corr[2]:L_Rho_H ~ lkj_corr_cholesky( 4 ),
        vector[2]:sigma_H ~ exponential(1),
        # fixed priors
        a ~ beta(4,4),
        g ~ exponential(0.5),
        c(b1,b2) ~ normal(0,0.5),
        gq> matrix[2,2]:Rho_H <<- Chol_to_Corr( L_Rho_H )
    ) , data=dat , chains=4 , cores=4 , iter=4000 )
precis(m2,3,pars=c("a","g","b1","b2","sigma_H"))
```

```
            mean   sd  5.5% 94.5% n_eff Rhat4
a           0.90 0.04  0.83  0.96  3671  1.00
g           8.11 2.22  4.99 11.96  1211  1.00
b1          2.23 0.12  2.03  2.42   648  1.01
b2          0.05 0.12 -0.16  0.22  1411  1.00
sigma_H[1] 0.17 0.08  0.03  0.30   422  1.01
sigma_H[2] 0.60 0.09  0.47  0.74   545  1.01
```

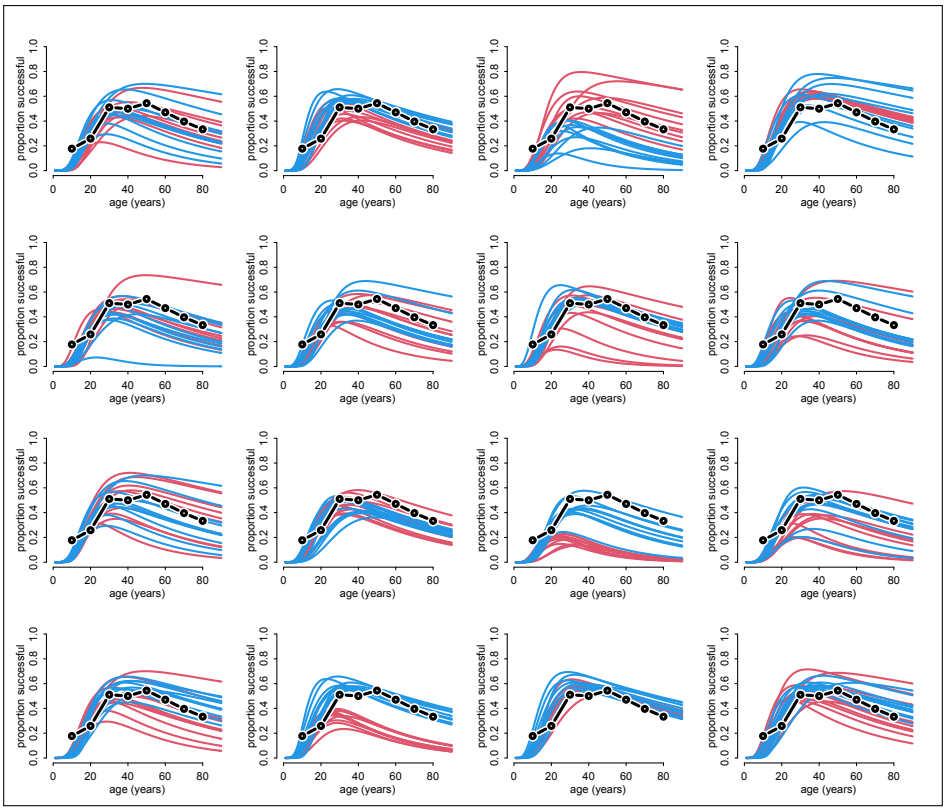
The variance components suggest that there is little variation in β_1 but more in β_2. So hunters differ more in their decline rates than their increase rates. Of course a faster decline rate will lower the entire curve for a hunter, so it effectively changes the maximum as well.

To get an impression for how much variation there is among hunters, let's plot some hunter functions. I'll make a grid of 16 plots, each showing a pair of hunters, with samples from the posterior shown in different colors for each hunter. We can compare within plots and across plots.

```
post <- extract.samples(m2)

# round age to nearest decade
Ar <- round( d$age / 10 ) * 10
Aseq <- c(10,20,30,40,50,60,70,80)
SA <- sapply( Aseq , function(a) mean(dat$S[Ar==a]) )

par(mfrow=c(4,4))
for (k in 1:16 ) {
    plot( Aseq , SA , ylim=c(0,1) , xlim=c(0,90) , type="b" ,
    lwd=3 , col=0 , xlab="age (years)" , ylab="proportion successful" )

    # plot nj random hunters
    cols <- c(2,4,5,6)
    nj <- 2
    hseq <- sample(1:dat$NH,size=nj)
    for ( j in hseq )
    for ( i in 1:10 ) with( post ,
        curve( a[i]*exp(-b2H[i,j]*x/80)*(1-exp(-b1H[i,j]*x/80))^g[i] ,
        add=TRUE , lwd=2 , from=1 , col=cols[which(hseq==j)] ) )
    points( Aseq , SA , type='b' , col="white" , lwd=6 )
    points( Aseq , SA , type='b' , col=1 , lwd=3 )
}#k
```

I show the plot on the next page. You can run the code above over and over again to sample new hunters. The overall impression is that hunters can vary quite a lot. But many of them are well represented by the mean.

The question asked if there is more variation by age or by hunter. The easy answer is that there is lots of both. But a principled answer (that is harder) is to compute



the variation in success across age, averaged over hunters, and compare it to the variation across hunters, averaged over age.

This is a subtle kind of calculation to do, because just using every age in the data is a bit weird—many hunters will never live to 80, so weighting the estimates at 80 years old the same as those at 20 years old seems wrong. Indeed it is wrong, if the goal is to say how much empirical realized variation in success owes to age differences or to other individual differences. Suppose for example that there is only one hunter who is 90 in the community. But he is really really good. He still won't contribute much to variation in success as explained by age differences in the total sample, because he is just one person.

Let's do it the easy way first, weighting each age the same. This isn't ideal, but it is the simplest example. First I compute the variation across age for each hunter.

```
# compute variation across age for each hunter
vHA <- rep(NA,dat$NH)
Aseq <- 10:80
for ( i in 1:dat$NH ) {
    # compute variation across age for each sample from posterior
    # then average across samples
    # v has margins [samples,ages]
```

```
    v <- sapply( Aseq , function(x)
        with( post ,
            a*exp(-b2H[,i]*x/80)*(1-exp(-b1H[,i]*x/80))^g ) )
    # now average variation across ages
    vHA[i] <- mean( apply( v , 1 , var ) )
}#i
```

Now the variation across hunters at each age.

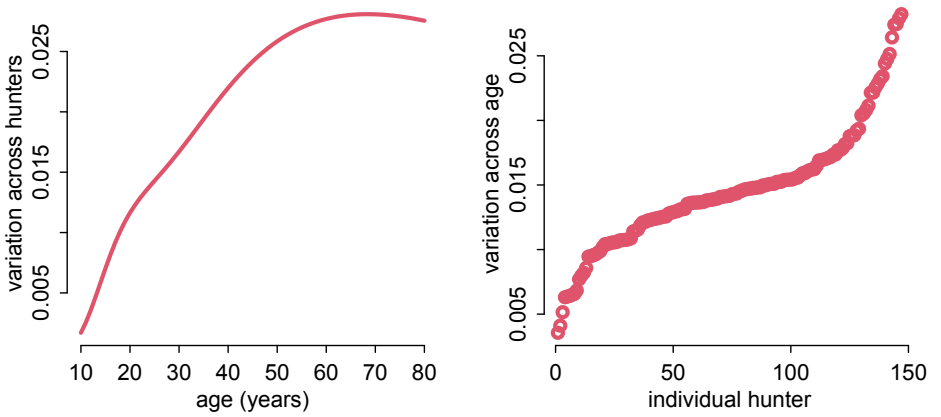
```
# variation across all hunters at each age
vAH <- rep(NA,length(Aseq))
for ( j in 1:length(Aseq) ) {
    # variation at age j across all hunters
    # v has margins [samples,hunters]
    v <- sapply( 1:dat$NH , function(i)
        with( post ,
            a*exp(-b2H[,i]*Aseq[j]/80)*(1-exp(-b1H[,i]*Aseq[j]/80))^g ) )
    # average variance across individuals
    vAH[j] <- mean( apply( v , 1 , var ) )
}#j
```

And we can compare these calculations directly:

```
# average across hunters (of variation across age)
mean(vHA)
# average across age (of variation across hunters)
mean(vAH)
```

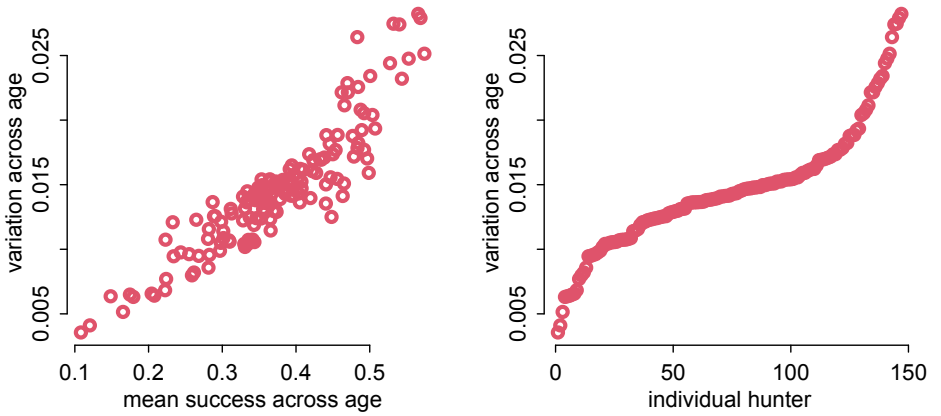
```
> mean(vHA)
[1] 0.01458467
> mean(vAH)
[1] 0.02096086
```

So the variation across ages is about 40% greater than that across individuals. Let's plot the elements of these calculations, to get a sense of what contributes to this difference.



The left shows the average variation at each age across hunters. Differences across hunters are greatest late in life, which is consistent with the variance components showing greater variation for the decline rates. Since late ages contribute the most variation across hunters, if we redid this comparison discounting later ages, since they appear less in the community, then the variation explained by individual differences might be more similar or even greater than the variation across age.

The right is each hunter on the horizontal and his posterior mean variation across ages. Some hunters change a lot over time, especially the good ones. This is easier to see if I plot the variation across age against the mean success across age:



The best hunters also vary the most over a lifetime.

3. The first problem is how to incorporate duration into the function relating age to success. To remind you, this is our success function:

$$p(A) = \alpha(1 - \exp(-\beta_1 A))^\gamma \exp(-\beta_2 A)$$

One approach to incorporating duration is to suppose the above is the function for some standard duration and then adjust it proportionally for shorter or longer trips.

A trip of duration zero should have zero chance of success. And the maximum chance of success can't exceed 1. But an infinitely long trip should have a 100% chance of success.

We can accomplish this by thinking (!) about what the function $p(A)$ means. Suppose it is defined for a standard duration of $L = 1$. Then the probability of success means the probability of capturing at least one animal in the duration $L = 1$. So we can conceive of the hunting process as having some rate of capture, and so for shorter or longer durations, we can model how the probability changes. Specifically, suppose animals are captured as a Poisson process, with a rate

$$\theta = L^{\lambda} p(A)$$

The duration L here is an *exposure* that defines the observation window for events to occur. The parameter λ defines the rate of diminishing returns on duration. And $p(A)$ is the base rate of events. Now what we need is the probability of success, which is the probability of at least one animal. The Poisson distribution is defined as:

$$\Pr(Y = y|\theta) = \frac{\theta^y \exp(-\theta)}{y!}$$

where $\theta = L^{\lambda} p(A)$ in our example. So the probability of zero events after time L is:

$$\Pr(Y = 0|\theta) = \frac{\theta^0 \exp(-\theta)}{0!} = \exp(-\theta)$$

And so the probability of at least one animal is simply:

$$\Pr(Y > 0|\theta) = 1 - \Pr(Y = 0|\theta) = 1 - \exp(-\theta)$$

And that's what we need to incorporate L into our model, let the probability of success now be:

$$p(A, L) = 1 - \exp(-L^{\lambda} f(A))$$
$$f(A) = (1 - \exp(-\beta_1 A))^{\gamma} \exp(-\beta_2 A)$$

Notice that I've removed the parameter α from the model, because it would be redundant. The scaling of duration L through λ adjust the height of the curve now.

In coding this into the model, I will express L on the log scale. That way we can do the missing data imputation more easily. Why? Because missing data are parameters, and parameters on the log scale are usually easier to sample than parameters with hard constraints. So this gives us a new model formula:

```
flist3 <- alist(
    S ~ bernoulli(p),
p <- 1-exp( -exp(lambda*log_L) * f ),
f <- exp(-b2H[H]*A2)*(1-exp(-b1H[H]*A2))^g,
    # centered varying effects
    transpars> vector[NH]:b1H <<- exp(b1+V[1:NH,1]),
    transpars> vector[NH]:b2H <<- exp(b2+V[1:NH,2]),
    # non-centered varying effects
    transpars> matrix[NH,2]:V <-
```

```
        compose_noncentered( sigma_H , L_Rho_H , Z ),
    matrix[2,NH]:Z ~ normal( 0 , 1 ),
    cholesky_factor_corr[2]:L_Rho_H ~ lkj_corr_cholesky( 4 ),
    vector[2]:sigma_H ~ exponential(1),
# duration prior
log_L ~ normal(muL,sigmaL),
muL ~ normal(-1,0.25),
sigmaL ~ exponential(2),
    # fixed priors
lambda ~ exponential(1),
    g ~ exponential(0.5),
    c(b1,b2) ~ normal(0,0.5),
    gq> matrix[2,2]:Rho_H <<- Chol_to_Corr( L_Rho_H )
)
```

The new bits are pulled out to the left, to make them easier to see. Inside the `p` definition, I have $\exp(\lambda \log L) = L^\lambda$. And then down lower the distribution for the $\log L$ is defined. This is a likelihood when L is observed and a prior when not.

Let's fit the model on complete cases first. It will be past and painless.

```
L = d$hours / max(d$hours,na.rm=TRUE)
dat$log_L <- log(L)

ccidx <- which(!is.na(dat$log_L))
datcc <- list(
    S = dat$S[ccidx],
    A2 = dat$A2[ccidx],
    H = dat$H[ccidx],
    NH = dat$NH,
    log_L = dat$log_L[ccidx]
)

m3cc <- ulam( flist3 , data=datcc , chains=4 , cores=4 , iter=1000 )
```

Let's fit the same model to the full sample now, and then we'll compare the models.

```
m3 <- ulam( flist3 , data=dat , chains=4 , cores=4 ,
    warmup=1000 , iter=4000 )
precis(m3)
precis(m3cc)
```

```
> precis(m3)
             mean   sd  5.5% 94.5% n_eff Rhat4
b1           2.41 0.11  2.23  2.57  2028     1
b2          -0.61 0.19 -0.93 -0.32  3630     1
g           11.18 3.01  6.86 16.43  3333     1
```

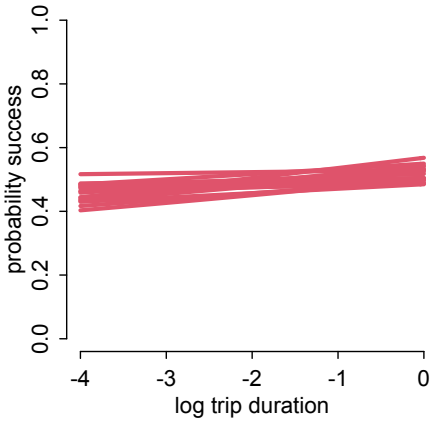
```
lambda      0.05 0.03  0.01  0.10  4981     1
sigma_H[1]  0.12 0.07  0.02  0.25  1466     1
sigma_H[2]  1.34 0.18  1.07  1.64  3155     1

> precis(m3cc)
            mean   sd  5.5% 94.5% n_eff Rhat4
b1          2.21 0.15  1.95  2.43   733  1.00
b2         -1.39 0.29 -1.88 -0.95  1533  1.00
g           7.73 2.53  4.14 12.04  1037  1.00
lambda      0.15 0.05  0.07  0.23  2333  1.00
sigma_H[1]  0.18 0.09  0.05  0.33   334  1.01
sigma_H[2]  1.35 0.28  0.95  1.83   965  1.00
```

So there are some obvious differences in the posterior distributions. Since there are only 2314 complete cases, but over 14-thousand total cases, that isn't surprising. But the variance components are very similar in both models. If you repeat the variance calculations from the previous problem, the basic pattern is the same: There is more variation across age than across individuals.

Really we should consider the causal structure of duration. It turns out that there isn't that much variation in trip duration in the data, and it isn't systematically related to age. So there isn't much opportunity for duration to influence our inference, except by adding some noise. And the `lambda` estimate indicates very quickly diminishing returns to duration.

We can see this clearly by plotting the posterior relationship between log duration and probability of success.

```
post <- extract.samples(m3)
f <- function(x,a=1,b1=8,b2=0.8,g=6) a*exp(-b2*x)*(1-exp(-b1*x))^g
plot( NULL , xlim=c(-4,0) , ylim=c(0,1) , xlab="log trip duration" ,
    ylab="probability success" )
for ( i in 1:20 )
with( post ,
curve( 1-exp( -exp(x)^lambda[i] *
    f(0.5,b1=exp(b1[i]),b2=exp(b2[i]),g=g[i]) ) ,
    add=TRUE , lwd=3 , col=2 )
)
```



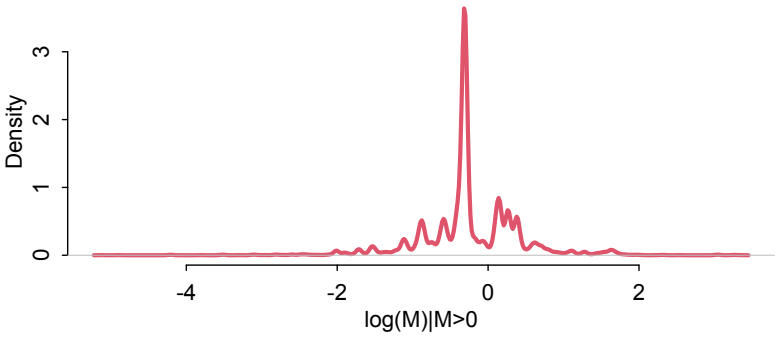
4-OPTIONAL CHALLENGE. If you plot the distribution of `kg.meat`—go ahead, do it—you'll see that it has a very long tail of large values. Most harvests are small. The average is about 7 kg. A few are many times larger. The largest is 190 kg. And more than half (7505 of 14364) of the trips returned zero. How can we model a distribution like this?

In an ideal world, we'd know more about the harvests. Hunters don't return from a trip with a bag full of "meat". They return with animals. We'd like to know how many animals and how much each weighed. Then we could model capture rates of each type of animal and its individual mass distribution to build up to the aggregate that we see in `kg.meat`.

But this isn't an ideal world. And that's okay. An ideal world would be boring.

To make progress, let's split off the non-zero values and try to model their distribution as a function of age and trip duration. Let's normalize the `kg.meat` values by dividing by the average non-zero value. Then I'll plot the log distribution:

```
dat$M <- d$kg.meat / mean(d$kg.meat[dat$S==1])
datcc$M <- dat$M[ccidx]
dens( log(dat$M[dat$S==1]) , lwd=3 , col=2 , xlab="log(M)|M>0" )
```



This isn't so bad to model as log-normal. Yeah those tails are very long. But keep in mind that the above is a mixture of distributions, because there is variation from hunters, age, and possibly also trip duration all folded together. Our job is never to model the aggregate outcome variable. Our job is always to model the conditional outcome variable. And besides, a Gaussian distribution is a device for estimating the mean and variance of some distribution. It works even when the frequency distribution isn't Gaussian. Of course, as I argued earlier in the course, if there are unmodeled sources of variation that produce outliers, it can help to use thicker tailed distributions, like a Student-t. But we can start with log-normal and see later if log-Student-t makes a difference.

A log-normal distribution is parameterized by the mean μ and standard deviation σ of the normal distribution that you exp() to get the observed distribution. The mean of a log-normal is $\exp(\mu + \sigma^2/2)$, because reasons, where the variance σ^2 effectively stretches out the long tail and increases the mean. We can tie our harvest rate function from the previous problem to the μ parameter. But we have to make sure it is $\exp(\mu)$ that is tied to it, not μ. So we can define:

$$\mu_i = \log(\alpha L_i^\lambda f(A_i)) = \log(\alpha) + \lambda \log(L_i) + \log(f(A_i))$$

The f function is unchanged. It still gives us the base rate of harvests. And then the duration L scales that rate to give us an expected mean. I added the intercept α because μ is not constrained to be below 1. We are not modeling probability anymore, but a harvest, and 1 is just the empirical mean.

All together now, here's the code:

```
flist4 <- alist(
    M|S==1 ~ dlnorm(mu,tau),
    mu <- a + lambda*log_L + log(f),
    f <- exp(-b2H[H]*A2)*(1-exp(-b1H[H]*A2))^g,
    tau ~ exponential(1),
    # centered varying effects
    transpars> vector[NH]:b1H <<- exp(b1+V[1:NH,1]),
    transpars> vector[NH]:b2H <<- exp(b2+V[1:NH,2]),
    # non-centered varying effects
    transpars> matrix[NH,2]:V <-
        compose_noncentered( sigma_H , L_Rho_H , Z ),
    matrix[2,NH]:Z ~ normal( 0 , 1 ),
    cholesky_factor_corr[2]:L_Rho_H ~ lkj_corr_cholesky( 4 ),
    vector[2]:sigma_H ~ exponential(1),
    # duration prior
    log_L ~ normal(muL,sigmaL),
    muL ~ normal(0,0.25),
    sigmaL ~ exponential(2),
    # fixed priors
    a ~ normal(0,0.5),
    lambda ~ exponential(1),
    g ~ exponential(0.5),
```

```
    c(b1,b2) ~ normal(0,0.5),
    gq> matrix[2,2]:Rho_H <<- Chol_to_Corr( L_Rho_H )
)
```

Note the trick in the first line where I make M conditional on $S = 1$. So the log-normal only triggers when $M > 0$. Then the line for μ is defined on the log scale. We could improve the model by also doing f on the log scale. In that case, the μ and f lines become:

```
    mu <- a + lambda*log_L + log_f,
    log_f <- (-b2H[H]*A2) + g*log(1-exp(-b1H[H]*A2)),
```

But it runs fine as is. Everything else in the model is the same as in the previous problem.

Let's run the model and peek at the variance components:

```
m4 <- ulam( flist4 , data=dat , chains=4 , cores=4 , warmup=1000 , iter=4000 )
precis(m4,3,pars=c("sigma_H"))
```

```
           mean   sd 5.5% 94.5% n_eff Rhat4
sigma_H[1] 0.32 0.06 0.24  0.42  5509     1
sigma_H[2] 0.24 0.10 0.10  0.40  1939     1
```

For success, there was more variation in the second component, the decline, but now the estimates are comparable.

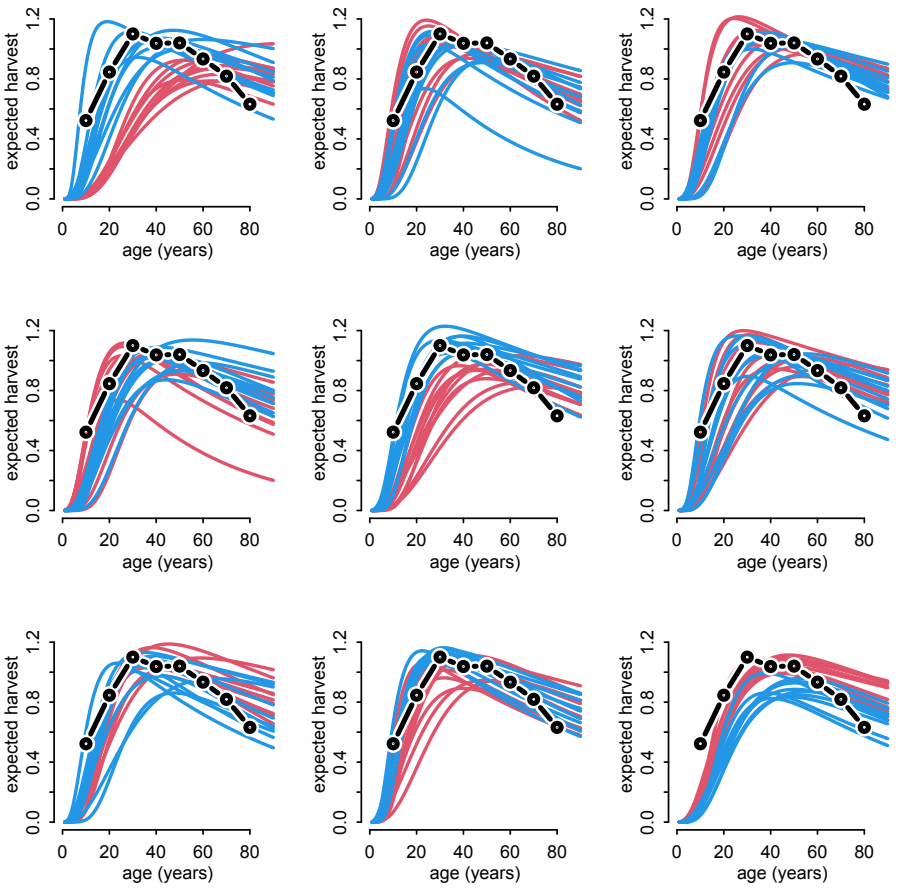
Let's look at the posterior predictions again. This time, we need to set the trip duration as well. I'll set it near the mean. And we need to include the variance `tau` in the calculation, which makes it a bit messy. Here's the code:

```
# round age to nearest decade
Ar <- round( d$age / 10 ) * 10
Aseq <- c(10,20,30,40,50,60,70,80)
MA <- sapply( Aseq , function(a) mean(dat$M[dat$S==1 & Ar==a]) )

post <- extract.samples(m4)
LL <- log(0.45)
par(mfrow=c(3,3))
for (k in 1:9 ) {
    plot( Aseq , MA , ylim=c(0,1.3) , xlim=c(0,90) , type="b" , lwd=3 ,
        col=0 , xlab="age (years)" , ylab="expected harvest" )

    # plot nj random hunters
    cols <- c(2,4,5,6)
    nj <- 2
    hseq <- sample(1:dat$NH,size=nj)
    for ( j in hseq )
```

```
    for ( i in 1:10 ) with( post ,
        curve(
                exp(
                    a[i] + lambda[i]*LL +
                    (-b2H[i,j]*x/80) + g[i]*log(1-exp(-b1H[i,j]*x/80))
                    + tau[i]^2/2
                ),
            add=TRUE , lwd=2 , from=1 , col=cols[which(hseq==j)] ) )
    points( Aseq , SA , type='b' , col="white" , lwd=6 )
    points( Aseq , SA , type='b' , col=1 , lwd=3 )
}#k
```

I show the plot on the next page. You can run the code many times to get an impression of the variation both across hunters and by age. You could also compute the variance comparisons again. And if you look at the relationship with trip duration, you'll see again that it doesn't matter much.

To understand what is going on with these data requires some thinking about how trip duration and harvest size are related to strategic aspects of hunting. For example, it may be that trip duration is rather constrained by other activities, reducing its variation. And what variation there is could be caused by success—trips that succeed early may end earlier, especially if hunters are limited in how much they can carry back to camp/town. A better way to approach data like these may be to consider success and duration to be joint outcomes of a dynamic encounter model. Such models are used in behavioral ecology, but I've never seen them applied to human production data. It would surely be a big project.


expected harvest
0.0
0.4
0.8
1.2
0
20
40
60
80
age (years)