tadpoles

Chris Parrish

June 9, 2016

Contents

tadpoles

references:
- McElreath, Statistical Rethinking, chap 12, p.355 - reedfrogs, Google images

```
library(rethinking)
library(ggplot2)
```

tadpoles in tanks

data

```
## R code 12.1
data(reedfrogs)
d <- reedfrogs
str(d)
```

```
## 'data.frame':    48 obs. of  5 variables:
##  $ density : int  10 10 10 10 10 10 10 10 10 10 ...
##  $ pred    : Factor w/ 2 levels "no","pred": 1 1 1 1 1 1 1 1 2 2 ...
##  $ size    : Factor w/ 2 levels "big","small": 1 1 1 1 2 2 2 2 1 1 ...
##  $ surv    : int  9 10 7 10 9 9 10 9 4 9 ...
```

```
##  $ propsurv: num  0.9 1 0.7 1 0.9 0.9 1 0.9 0.4 0.9 ...
```

```
## R code 12.2
# make the tank cluster variable
d$tank <- 1:nrow(d)
d <- d[ , c("density", "surv", "propsurv", "tank")]
str(d)
```

```
## 'data.frame':    48 obs. of  4 variables:
##  $ density : int  10 10 10 10 10 10 10 10 10 10 ...
##  $ surv    : int  9 10 7 10 9 9 10 9 4 9 ...
##  $ propsurv: num  0.9 1 0.7 1 0.9 0.9 1 0.9 0.4 0.9 ...
##  $ tank    : int  1 2 3 4 5 6 7 8 9 10 ...
```

first model

One level: the vector of parameters has a prior

$$
\begin{aligned}
s_i &\sim Binomial(n_i, p_i) \\
logit(p_i) &= \alpha_{TANK[i]} \\
\alpha_{TANK[i]} &\sim Normal(0, 5)
\end{aligned}
$$

stan

```
# fit
m12.1 <- map2stan(
    alist(
        surv ~ dbinom( density , p ) ,
        logit(p) <- a_tank[tank] ,
        a_tank[tank] ~ dnorm( 0 , 5 )
    ),
    data=d )
```

```
##
## SAMPLING FOR MODEL 'surv ~ dbinom(density, p)' NOW (CHAIN 1).
##
## Chain 1, Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1, Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1, Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1, Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1, Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1, Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1, Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1, Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1, Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1, Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1, Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1, Iteration: 2000 / 2000 [100%]  (Sampling)
##  Elapsed Time: 0.13934 seconds (Warm-up)
##                0.142025 seconds (Sampling)
##                0.281365 seconds (Total)
##
```

```
##
## SAMPLING FOR MODEL 'surv ~ dbinom(density, p)' NOW (CHAIN 1).
## WARNING: No variance estimation is
##          performed for num_warmup < 20
##
##
## Chain 1, Iteration: 1 / 1 [100%]  (Sampling)
##  Elapsed Time: 2e-06 seconds (Warm-up)
##                7.2e-05 seconds (Sampling)
##                7.4e-05 seconds (Total)
```

```
## Computing WAIC
```

```
## Constructing posterior predictions
```

```
## [ 100 / 1000 ]
[ 200 / 1000 ]
[ 300 / 1000 ]
[ 400 / 1000 ]
[ 500 / 1000 ]
[ 600 / 1000 ]
[ 700 / 1000 ]
[ 800 / 1000 ]
[ 900 / 1000 ]
[ 1000 / 1000 ]
```

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

```
# plot(m12.1)
```

second model

Two levels: (1) the vector of parameters has a prior, (2) whose parameters have priors.

$$
\begin{aligned}
s_i &\sim Binomial(n_i, p_i) \\
logit(p_i) &= \alpha_{TANK[i]} \\
\alpha_{TANK[i]} &\sim Normal(a, \sigma) \\
a &\sim Normal(0, 1) \\
\sigma &\sim HalfCauchy(0, 1)
\end{aligned}
$$

stan

```
## R code 12.3
m12.2 <- map2stan(
    alist(
        surv ~ dbinom( density , p ) ,
        logit(p) <- a_tank[tank] ,
        a_tank[tank] ~ dnorm( a , sigma ) ,
        a ~ dnorm(0,1) ,
        sigma ~ dcauchy(0,1)
    ), data=d , iter=4000 , chains=4 )
```

```
##
## SAMPLING FOR MODEL 'surv ~ dbinom(density, p)' NOW (CHAIN 1).
##
## Chain 1, Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 1, Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 1, Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 1, Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 1, Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 1, Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 1, Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 1, Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 1, Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 1, Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 1, Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 1, Iteration: 4000 / 4000 [100%]  (Sampling)
##  Elapsed Time: 0.219213 seconds (Warm-up)
##                0.184107 seconds (Sampling)
##                0.40332 seconds (Total)
##
##
## SAMPLING FOR MODEL 'surv ~ dbinom(density, p)' NOW (CHAIN 2).
##
## Chain 2, Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 2, Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 2, Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 2, Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 2, Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 2, Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 2, Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 2, Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 2, Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 2, Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 2, Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 2, Iteration: 4000 / 4000 [100%]  (Sampling)
##  Elapsed Time: 0.215057 seconds (Warm-up)
##                0.179775 seconds (Sampling)
##                0.394832 seconds (Total)
##
##
## SAMPLING FOR MODEL 'surv ~ dbinom(density, p)' NOW (CHAIN 3).
##
## Chain 3, Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 3, Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 3, Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 3, Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 3, Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 3, Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 3, Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 3, Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 3, Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 3, Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 3, Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 3, Iteration: 4000 / 4000 [100%]  (Sampling)
##  Elapsed Time: 0.21927 seconds (Warm-up)
```

```
##                0.196846 seconds (Sampling)
##                0.416116 seconds (Total)

## The following numerical problems occured the indicated number of times after warmup on chain 3

##                                                                                 count
## Exception thrown at line 17: normal_log: Scale parameter is 0, but must be > 0!     1

## When a numerical problem occurs, the Metropolis proposal gets rejected.

## However, by design Metropolis proposals sometimes get rejected even when there are no numerical prob

## Thus, if the number in the 'count' column is small, do not ask about this message on stan-users.

##
## SAMPLING FOR MODEL 'surv ~ dbinom(density, p)' NOW (CHAIN 4).
##
## Chain 4, Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 4, Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 4, Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 4, Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 4, Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 4, Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 4, Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 4, Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 4, Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 4, Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 4, Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 4, Iteration: 4000 / 4000 [100%]  (Sampling)
##  Elapsed Time: 0.217643 seconds (Warm-up)
##                0.188137 seconds (Sampling)
##                0.40578 seconds (Total)

## The following numerical problems occured the indicated number of times after warmup on chain 4

##                                                                                 count
## Exception thrown at line 17: normal_log: Scale parameter is 0, but must be > 0!     2

## When a numerical problem occurs, the Metropolis proposal gets rejected.

## However, by design Metropolis proposals sometimes get rejected even when there are no numerical prob

## Thus, if the number in the 'count' column is small, do not ask about this message on stan-users.

##
## SAMPLING FOR MODEL 'surv ~ dbinom(density, p)' NOW (CHAIN 1).
## WARNING: No variance estimation is
##          performed for num_warmup < 20
##
##
## Chain 1, Iteration: 1 / 1 [100%]  (Sampling)
##  Elapsed Time: 5e-06 seconds (Warm-up)
##                6.6e-05 seconds (Sampling)
##                7.1e-05 seconds (Total)

## Computing WAIC

## Constructing posterior predictions

## [ 800 / 8000 ]
[ 1600 / 8000 ]
```

```
[ 2400 / 8000 ]
[ 3200 / 8000 ]
[ 4000 / 8000 ]
[ 4800 / 8000 ]
[ 5600 / 8000 ]
[ 6400 / 8000 ]
[ 7200 / 8000 ]
[ 8000 / 8000 ]
## Aggregated binomial counts detected. Splitting to 0/1 outcome for WAIC calculation.
```

```
# plot(m12.1)
```

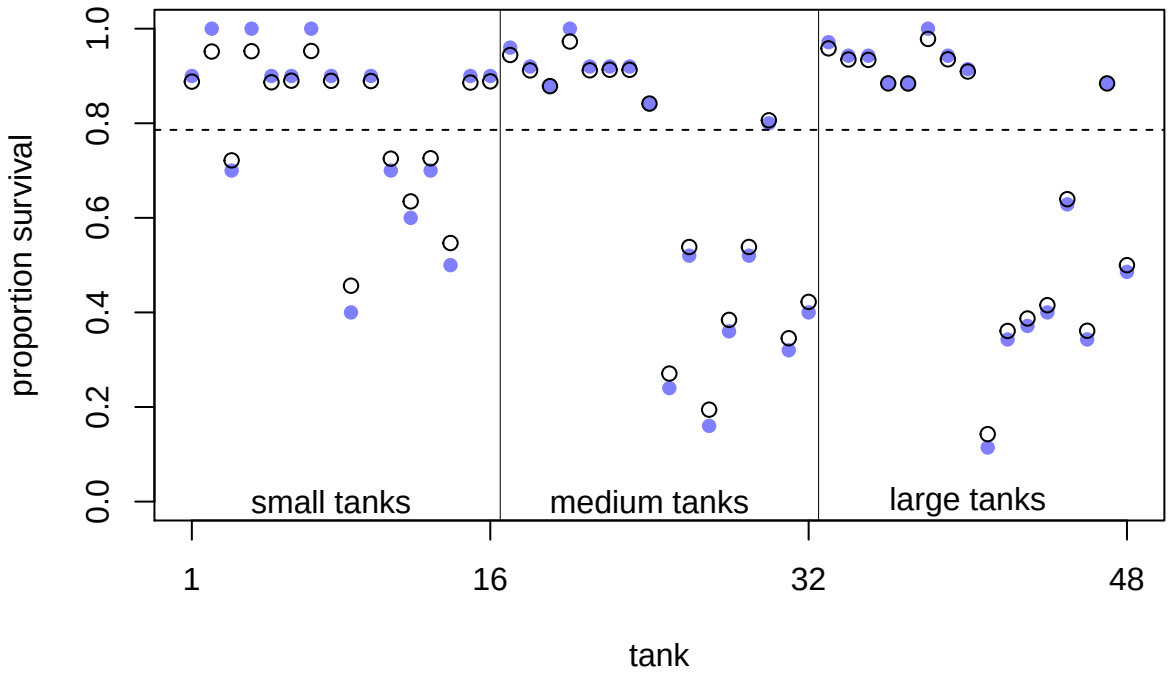
compare

```
## R code 12.4
compare( m12.1 , m12.2 )
```

```
##         WAIC pWAIC dWAIC weight    SE  dSE
## m12.2 1010.2  38.2   0.0      1 38.15   NA
## m12.1 1026.4  51.0  16.2      0 43.32 6.83
```

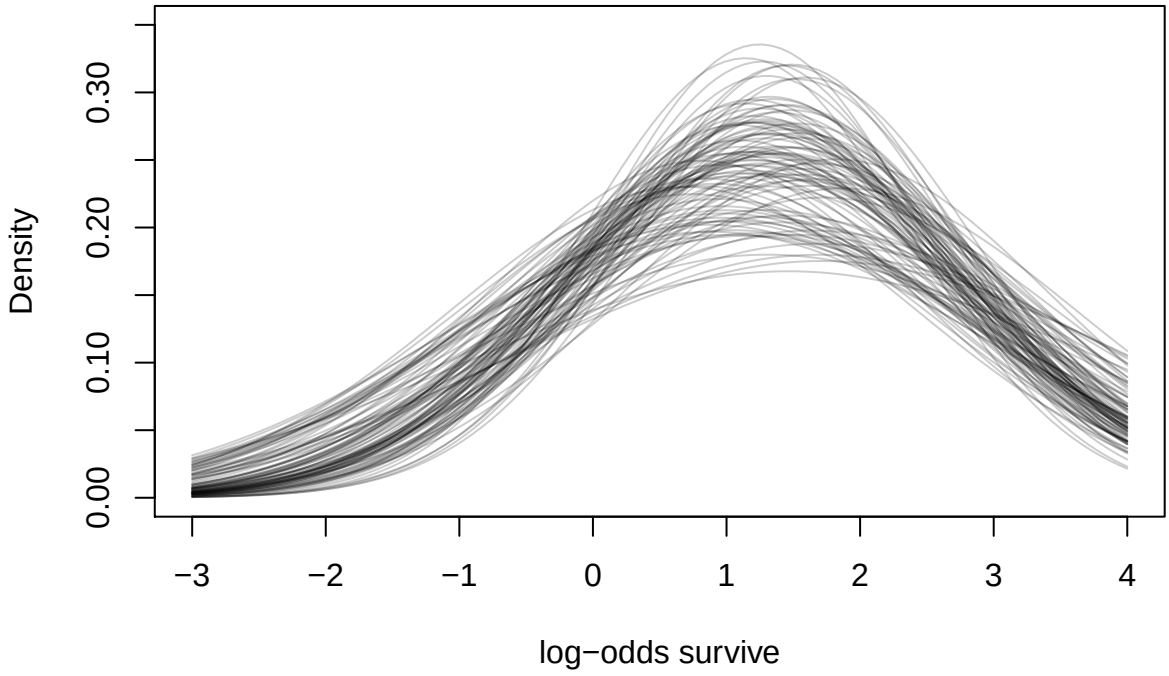
illustration

```
## R code 12.5
# extract Stan samples
post <- extract.samples(m12.2)

# compute median intercept for each tank
# also transform to probability with logistic
d$propsurv.est <- logistic( apply( post$a_tank , 2 , median ) )

# display raw proportions surviving in each tank
plot( d$propsurv , ylim=c(0,1) , pch=16 , xaxt="n" ,
    xlab="tank" , ylab="proportion survival" , col=rangi2 )
axis( 1 , at=c(1,16,32,48) , labels=c(1,16,32,48) )

# overlay posterior medians
points( d$propsurv.est )

# mark posterior median probability across tanks
abline( h=logistic(median(post$a)) , lty=2 )

# draw vertical dividers between tank densities
abline( v=16.5 , lwd=0.5 )
abline( v=32.5 , lwd=0.5 )
text( 8 , 0 , "small tanks" )
text( 16+8 , 0 , "medium tanks" )
text( 32+8 , 0 , "large tanks" )
```

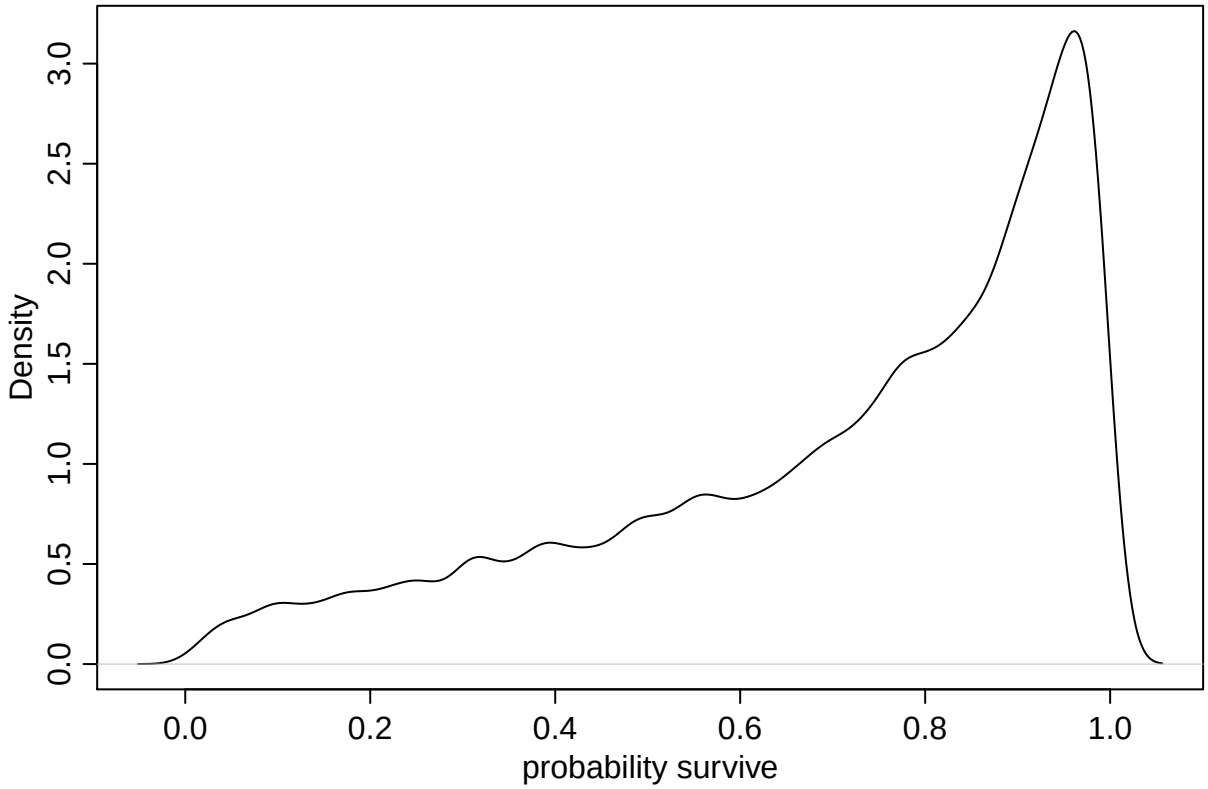


survival across tanks

```
## R code 12.6
# show first 100 populations in the posterior
plot( NULL , xlim=c(-3,4) , ylim=c(0,0.35) ,
    xlab="log-odds survive" , ylab="Density" )
for ( i in 1:100 )
    curve( dnorm(x,post$a[i],post$sigma[i]) , add=TRUE ,
    col=col.alpha("black",0.2) )
```



```r
# sample 8000 imaginary tanks from the posterior distribution
sim_tanks <- rnorm( 8000 , post$a , post$sigma )

# transform to probability and visualize
dens( logistic(sim_tanks) , xlab="probability survive" )
```



tadpoles in ponds

model

Two levels: (1) the vector of parameters has a prior, (2) whose parameters have priors.

$$
\begin{aligned}
s_i &\sim Binomial(n_i, p_i) \\
logit(p_i) &= \alpha_{TANK[i]} \\
\alpha_{TANK[i]} &\sim Normal(a, \sigma) \\
a &\sim Normal(0, 1) \\
\sigma &\sim HalfCauchy(0, 1)
\end{aligned}
$$

initialize parameters

```r
## R code 12.7
a <- 1.4
sigma <- 1.5
```

```
nponds <- 60
ni <- as.integer( rep( c(5,10,25,35) , each=15 ) )
```

simulate values

```
## R code 12.8
a_pond <- rnorm( nponds , mean=a , sd=sigma )
str(a_pond)
```

```
##  num [1:60] -0.687 2.347 1.98 3.594 2.117 ...
```

```
## R code 12.9
dsim <- data.frame( pond=1:nponds , ni=ni , true_a=a_pond )
str(dsim)
```

```
## 'data.frame':    60 obs. of  3 variables:
##  $ pond  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ ni    : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ true_a: num  -0.687 2.347 1.98 3.594 2.117 ...
```

simulate survivors

```
## R code 12.11
dsim$si <- rbinom( nponds , prob=logistic(dsim$true_a) , size=dsim$ni )
```

compute no-pooling estimates

```
## R code 12.12
dsim$p_nopool <- dsim$si / dsim$ni
```

compute partial-pooling estimates

stan

```
## R code 12.13
m12.3 <- map2stan(
    alist(
        si ~ dbinom( ni , p ),
        logit(p) <- a_pond[pond],
        a_pond[pond] ~ dnorm( a , sigma ),
        a ~ dnorm(0,1),
        sigma ~ dcauchy(0,1)
    ),
    data=dsim , iter=1e4 , warmup=1000 )
```

```
##
## SAMPLING FOR MODEL 'si ~ dbinom(ni, p)' NOW (CHAIN 1).
##
## Chain 1, Iteration:    1 / 10000 [  0%]  (Warmup)
```

```
## Chain 1, Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 1, Iteration: 1001 / 10000 [ 10%]  (Sampling)
## Chain 1, Iteration: 2000 / 10000 [ 20%]  (Sampling)
## Chain 1, Iteration: 3000 / 10000 [ 30%]  (Sampling)
## Chain 1, Iteration: 4000 / 10000 [ 40%]  (Sampling)
## Chain 1, Iteration: 5000 / 10000 [ 50%]  (Sampling)
## Chain 1, Iteration: 6000 / 10000 [ 60%]  (Sampling)
## Chain 1, Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 1, Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 1, Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 1, Iteration: 10000 / 10000 [100%]  (Sampling)
##  Elapsed Time: 0.154813 seconds (Warm-up)
##                1.28879 seconds (Sampling)
##                1.4436 seconds (Total)
##
##
## SAMPLING FOR MODEL 'si ~ dbinom(ni, p)' NOW (CHAIN 1).
## WARNING: No variance estimation is
##          performed for num_warmup < 20
##
##
## Chain 1, Iteration: 1 / 1 [100%]  (Sampling)
##  Elapsed Time: 4e-06 seconds (Warm-up)
##                8.2e-05 seconds (Sampling)
##                8.6e-05 seconds (Total)

## Computing WAIC

## Constructing posterior predictions

## [ 900 / 9000 ]
[ 1800 / 9000 ]
[ 2700 / 9000 ]
[ 3600 / 9000 ]
[ 4500 / 9000 ]
[ 5400 / 9000 ]
[ 6300 / 9000 ]
[ 7200 / 9000 ]
[ 8100 / 9000 ]
[ 9000 / 9000 ]

## Aggregated binomial counts detected. Splitting to 0/1 outcome for WAIC calculation.
## R code 12.14
precis(m12.3,depth=2)

##             Mean StdDev lower 0.89 upper 0.89 n_eff Rhat
## a_pond[1]   0.04   0.83      -1.24       1.42  9000    1
## a_pond[2]   2.72   1.23       0.86       4.64  9000    1
## a_pond[3]   2.71   1.20       0.76       4.54  9000    1
## a_pond[4]   2.74   1.22       0.79       4.60  6878    1
## a_pond[5]   1.60   0.99       0.04       3.16  9000    1
## a_pond[6]   0.75   0.87      -0.64       2.11  9000    1
## a_pond[7]  -0.65   0.88      -2.00       0.76  9000    1
## a_pond[8]   0.75   0.87      -0.61       2.15  8227    1
## a_pond[9]  -0.66   0.87      -2.04       0.71  9000    1
## a_pond[10]  0.75   0.88      -0.70       2.10  8507    1
```

```
## a_pond[11]  2.72   1.23       0.80       4.65  9000    1
## a_pond[12]  0.75   0.88      -0.62       2.16  7487    1
## a_pond[13]  2.72   1.23       0.82       4.66  9000    1
## a_pond[14]  0.76   0.87      -0.63       2.12  9000    1
## a_pond[15]  2.73   1.23       0.78       4.60  6903    1
## a_pond[16]  2.18   0.88       0.79       3.54  9000    1
## a_pond[17]  1.53   0.76       0.32       2.69  9000    1
## a_pond[18]  2.18   0.90       0.76       3.57  9000    1
## a_pond[19]  2.17   0.89       0.79       3.56  9000    1
## a_pond[20]  3.15   1.16       1.29       4.88  9000    1
## a_pond[21] -0.16   0.63      -1.15       0.86  9000    1
## a_pond[22]  3.13   1.12       1.35       4.80  9000    1
## a_pond[23]  3.13   1.13       1.39       4.86  9000    1
## a_pond[24] -0.54   0.64      -1.53       0.50  9000    1
## a_pond[25]  3.13   1.14       1.26       4.77  6380    1
## a_pond[26]  1.02   0.68      -0.13       2.03  9000    1
## a_pond[27]  0.23   0.62      -0.74       1.22  9000    1
## a_pond[28]  3.12   1.12       1.39       4.85  9000    1
## a_pond[29]  2.17   0.89       0.84       3.61  9000    1
## a_pond[30]  2.19   0.89       0.76       3.54  9000    1
## a_pond[31]  2.41   0.67       1.35       3.43  9000    1
## a_pond[32]  2.94   0.79       1.75       4.23  9000    1
## a_pond[33] -0.15   0.40      -0.77       0.50  9000    1
## a_pond[34]  0.66   0.42      -0.04       1.31  9000    1
## a_pond[35]  2.94   0.81       1.62       4.12  9000    1
## a_pond[36]  3.73   1.04       2.10       5.33  6924    1
## a_pond[37] -0.99   0.45      -1.68      -0.26  9000    1
## a_pond[38]  2.95   0.81       1.66       4.16  9000    1
## a_pond[39]  1.02   0.45       0.32       1.73  8519    1
## a_pond[40]  0.01   0.39      -0.61       0.66  9000    1
## a_pond[41]  2.03   0.60       1.03       2.93  7934    1
## a_pond[42]  0.83   0.43       0.12       1.49  9000    1
## a_pond[43]  3.73   1.07       2.04       5.32  6261    1
## a_pond[44]  1.22   0.47       0.47       1.95  9000    1
## a_pond[45]  1.72   0.54       0.90       2.60  6575    1
## a_pond[46] -1.10   0.39      -1.68      -0.46  9000    1
## a_pond[47]  0.70   0.36       0.13       1.26  9000    1
## a_pond[48] -0.69   0.36      -1.27      -0.14  9000    1
## a_pond[49]  2.07   0.52       1.23       2.86  9000    1
## a_pond[50] -0.10   0.34      -0.63       0.43  9000    1
## a_pond[51] -0.33   0.34      -0.88       0.19  9000    1
## a_pond[52]  1.12   0.39       0.50       1.73  9000    1
## a_pond[53]  3.23   0.78       2.01       4.45  9000    1
## a_pond[54]  2.08   0.52       1.26       2.88  9000    1
## a_pond[55] -0.21   0.34      -0.76       0.34  9000    1
## a_pond[56]  2.36   0.57       1.50       3.29  7744    1
## a_pond[57] -0.57   0.36      -1.14       0.00  7750    1
## a_pond[58]  2.73   0.66       1.68       3.73  9000    1
## a_pond[59]  1.64   0.45       0.93       2.35  9000    1
## a_pond[60]  3.96   1.00       2.33       5.42  4228    1
## a           1.45   0.24       1.06       1.83  5730    1
## sigma       1.62   0.23       1.27       1.97  2461    1
```

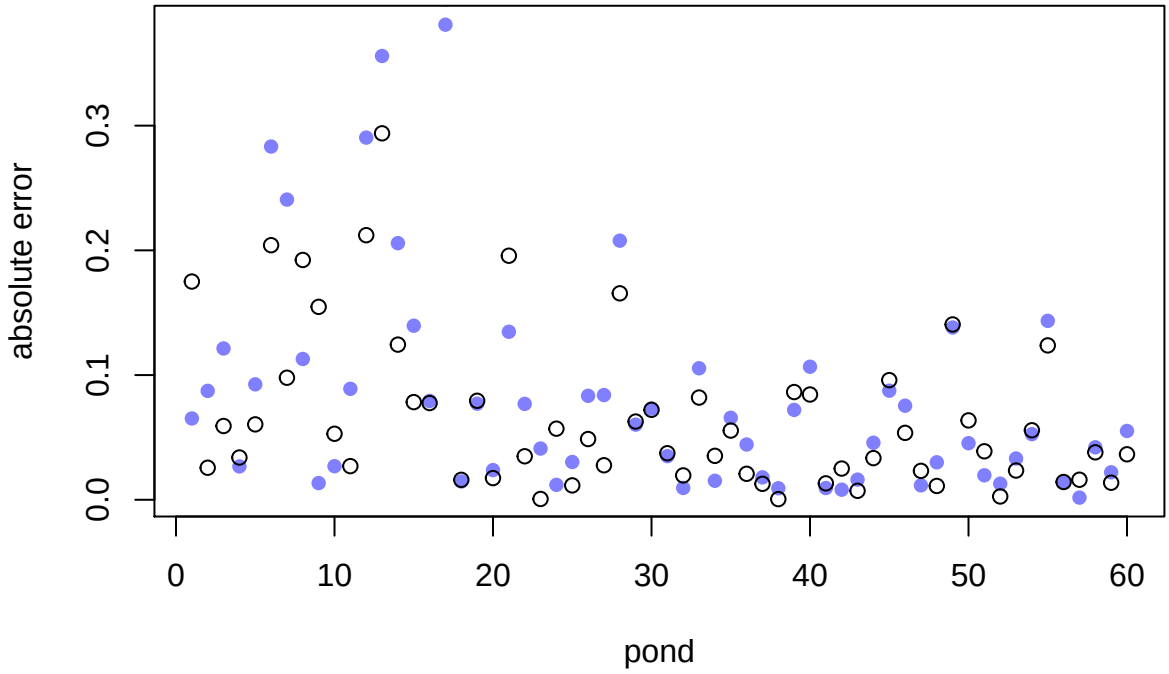
```
## R code 12.15
estimated.a_pond <- as.numeric( coef(m12.3)[1:60] )
dsim$p_partpool <- logistic( estimated.a_pond )

## R code 12.16
dsim$p_true <- logistic( dsim$true_a )
```

compute errors

```
## R code 12.17
nopool_error <- abs( dsim$p_nopool - dsim$p_true )
partpool_error <- abs( dsim$p_partpool - dsim$p_true )
```

plot results

```
## R code 12.18
plot( 1:60 , nopool_error , xlab="pond" , ylab="absolute error" ,
    col=rangi2 , pch=16 )
points( 1:60 , partpool_error )
```



reuse compiled model

```
## R code 12.19
a <- 1.4
sigma <- 1.5
nponds <- 60
ni <- as.integer( rep( c(5,10,25,35) , each=15 ) )
```

```
a_pond <- rnorm( nponds , mean=a , sd=sigma )
dsim <- data.frame( pond=1:nponds , ni=ni , true_a=a_pond )
dsim$si <- rbinom( nponds,prob=logistic( dsim$true_a ),size=dsim$ni )
dsim$p_nopool <- dsim$si / dsim$ni
newdat <- list(si=dsim$si,ni=dsim$ni,pond=1:nponds)
m12.3new <- map2stan( m12.3 , data=newdat , iter=1e4 , warmup=1000 )
```

```
##
## SAMPLING FOR MODEL 'si ~ dbinom(ni, p)' NOW (CHAIN 1).
##
## Chain 1, Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 1, Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 1, Iteration: 1001 / 10000 [ 10%]  (Sampling)
## Chain 1, Iteration: 2000 / 10000 [ 20%]  (Sampling)
## Chain 1, Iteration: 3000 / 10000 [ 30%]  (Sampling)
## Chain 1, Iteration: 4000 / 10000 [ 40%]  (Sampling)
## Chain 1, Iteration: 5000 / 10000 [ 50%]  (Sampling)
## Chain 1, Iteration: 6000 / 10000 [ 60%]  (Sampling)
## Chain 1, Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 1, Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 1, Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 1, Iteration: 10000 / 10000 [100%]  (Sampling)
##  Elapsed Time: 0.15198 seconds (Warm-up)
##                0.983912 seconds (Sampling)
##                1.13589 seconds (Total)
```

```
## The following numerical problems occured the indicated number of times after warmup on chain 1
```

```
##                                                                                          count
## Exception thrown at line 17: normal_log: Scale parameter is 0, but must be > 0!     1
```

```
## When a numerical problem occurs, the Metropolis proposal gets rejected.
```

```
## However, by design Metropolis proposals sometimes get rejected even when there are no numerical probl
```

```
## Thus, if the number in the 'count' column is small, do not ask about this message on stan-users.
```

```
##
## SAMPLING FOR MODEL 'si ~ dbinom(ni, p)' NOW (CHAIN 1).
## WARNING: No variance estimation is
##          performed for num_warmup < 20
##
##
## Chain 1, Iteration: 1 / 1 [100%]  (Sampling)
##  Elapsed Time: 3e-06 seconds (Warm-up)
##                6.5e-05 seconds (Sampling)
##                6.8e-05 seconds (Total)
```

```
## Computing WAIC
```

```
## Constructing posterior predictions
```

```
## [ 900 / 9000 ]
[ 1800 / 9000 ]
[ 2700 / 9000 ]
[ 3600 / 9000 ]
[ 4500 / 9000 ]
[ 5400 / 9000 ]
```

```
[ 6300 / 9000 ]
[ 7200 / 9000 ]
[ 8100 / 9000 ]
[ 9000 / 9000 ]
```

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