Multilevel should be default

- Defaults are powerful things
- Single-level regression is default
  - People justify multilevel models
- This is backwards
  - Multilevel estimates usually better
  - Should have to justify not using multilevel model

Goals

- Introduce multilevel models
- How *shrinkage* and *pooling* work
- Why they produce better estimates
- How to program with `ulam`
- Methods of plotting and comparing
- Open up more options

Multilevel models

- Usual use is to model clustering
  - Classrooms within schools
  - Students within classrooms
  - Grades within students
  - Questions within exams
- Repeat measures of units
- Imbalance in sampling
- “pseudoreplication”

Multilevel models

- Examples from earlier:
  - !Kung individuals in families
  - Species in clades
  - Nations in continents
  - Applicants in departments

Regularized intercepts

R code 13.2

```
# make the tank cluster variable
d$tank <- 1:nrow(d)

dat <- list(
    S = d$surv,
    N = d$density,
    tank = d$tank )

# approximate posterior
m13.1 <- ulam(
    alist(
        S ~ dbinom( N , p ) ,
        logit(p) <- a[tank] ,
        a[tank] ~ dnorm( 0 , 1.5 )
    ), data=dat , chains=4 , log_lik=TRUE )
```

$$S_i \sim \text{Binomial}(N_i, p_i)$$
$$\text{logit}(p_i) = \alpha_{\text{TANK}[i]}$$
$$\alpha_j \sim \text{Normal}(0, 1.5)$$

Terminology

- ***Varying intercepts*** also called ***random intercepts***
- Neither of these terms makes much sense
  - "random"? Sometimes associated with research design, but design irrelevant
  - Ordinary dummy variables also "vary" across clusters
- Distinctive because individual intercepts learn from one another
  - *mnestic*: opposite of *amnestic*

Adaptive regularization

$$
\begin{aligned}
S_i &\sim \text{Binomial}(N_i, p_i) \\
\text{logit}(p_i) &= \alpha_{\text{TANK}[i]} \\
\alpha_j &\sim \text{Normal}(\bar{\alpha}, \sigma) \\
\bar{\alpha} &\sim \text{Normal}(0, 1.5) \\
\sigma &\sim \text{Exponential}(1)
\end{aligned}
$$

varying intercepts ⟶ $\alpha_j \sim \text{Normal}(\bar{\alpha}, \sigma)$

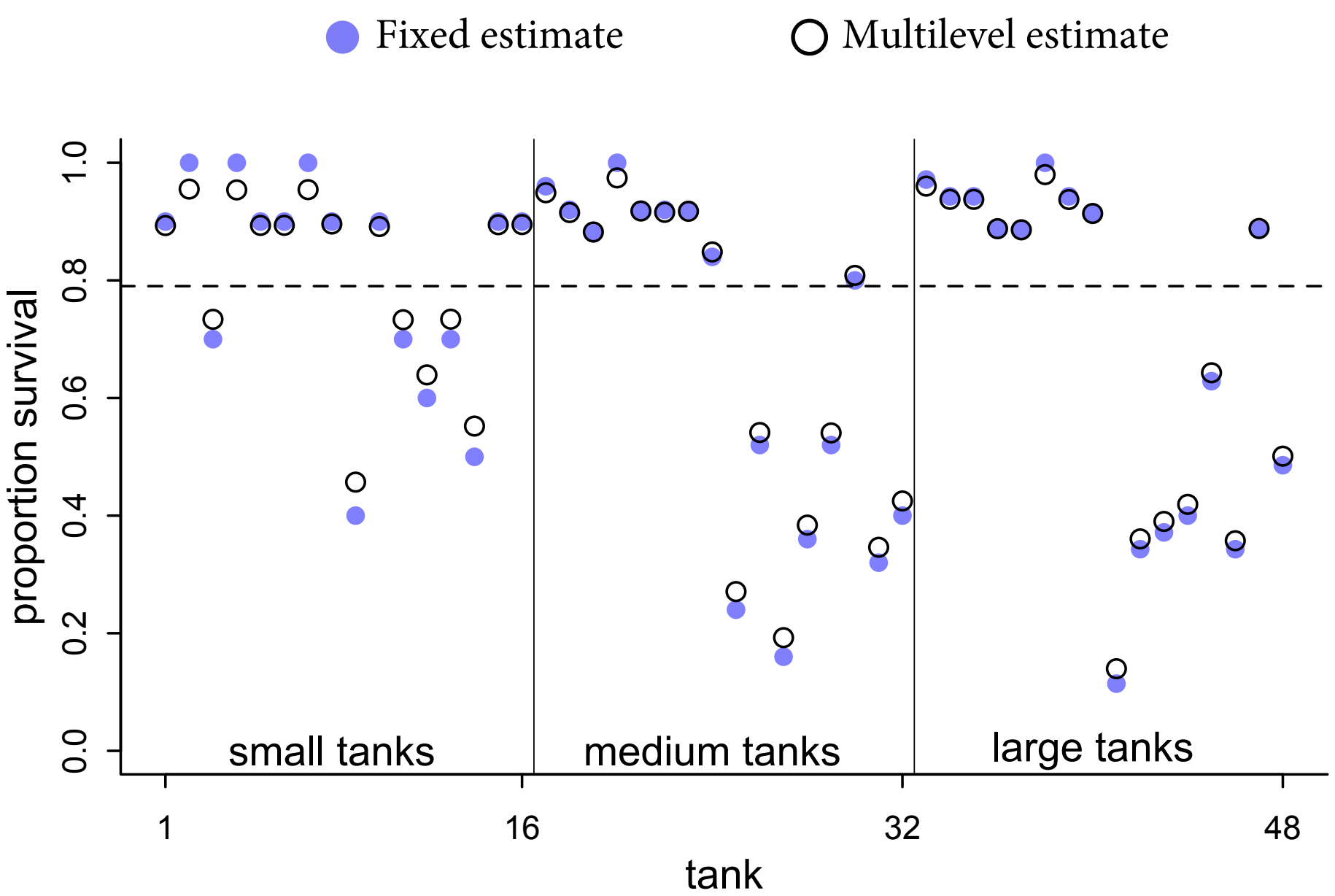
Survival across tanks has a *distribution*.
This *distribution* is the prior for each tank.
Distribution needs its own prior.

R code 13.4

```
compare( m13.1 , m13.2 )
```

```
       WAIC pWAIC dWAIC weight   SE  dSE
m13.2   202  21.7     0      1 7.35   NA
m13.1   213  24.8    11      0 4.71 3.58
```

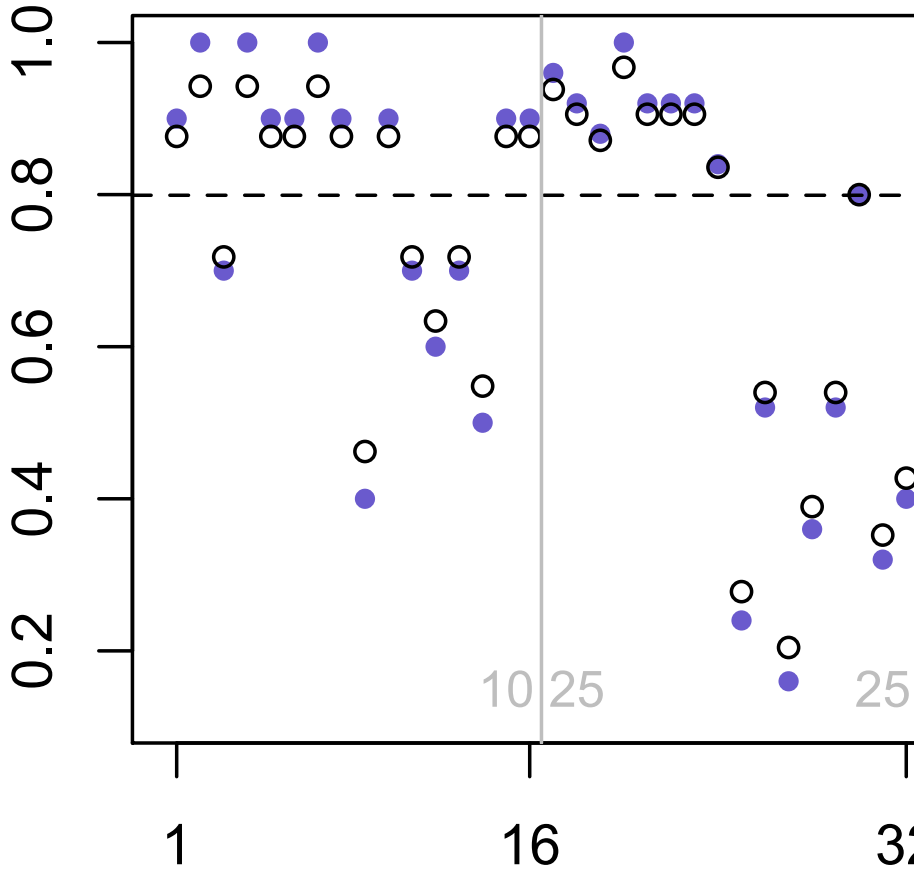
- m13.1: 48 parameters vs 25 effective
- m13.2: 50 parameters vs 22 effective
- Model with more parameters has fewer effective parameters
- Why? Ended up with stronger prior.



Don't expect predictions to match observations exactly. Instead expect ***shrinkage***.

Shrinkage

- Varying effect estimates *shrink* towards mean (a_bar)
- Further from mean, more shrinkage
- Fewer data in cluster, more shrinkage
- Same as regression to the mean, really
- Shrinkage results from *pooling* of information



Ulysses' Compass again

- Why are ***varying effects*** (partial pooling) more accurate than ***fixed effects*** (no pooling)?
- Grand mean: maximum underfitting
- Fixed effects: maximum overfitting
- Varying effects: adaptive regularization

Multilevel chimpanzees

$$
\begin{aligned}
L_i &\sim \text{Binomial}(1, p_i) \\
\text{logit}(p_i) &= \alpha_{\text{ACTOR}[i]} + \gamma_{\text{BLOCK}[i]} + \beta_{\text{TREATMENT}[i]} \\
\beta_j &\sim \text{Normal}(0, 0.5) \quad \text{, for } j = 1..4 \\
\alpha_j &\sim \text{Normal}(\bar{\alpha}, \sigma_\alpha) \quad \text{, for } j = 1..7 \\
\gamma_j &\sim \text{Normal}(0, \sigma_\gamma) \quad \text{, for } j = 1..6 \\
\bar{\alpha} &\sim \text{Normal}(0, 1.5) \\
\sigma_\alpha &\sim \text{Exponential}(1) \\
\sigma_\gamma &\sim \text{Exponential}(1)
\end{aligned}
$$

varying intercepts on actor ⟶ $\alpha_j \sim \text{Normal}(\bar{\alpha}, \sigma_\alpha)$, for $j = 1..7$

varying intercepts on block ⟶ $\gamma_j \sim \text{Normal}(0, \sigma_\gamma)$, for $j = 1..6$

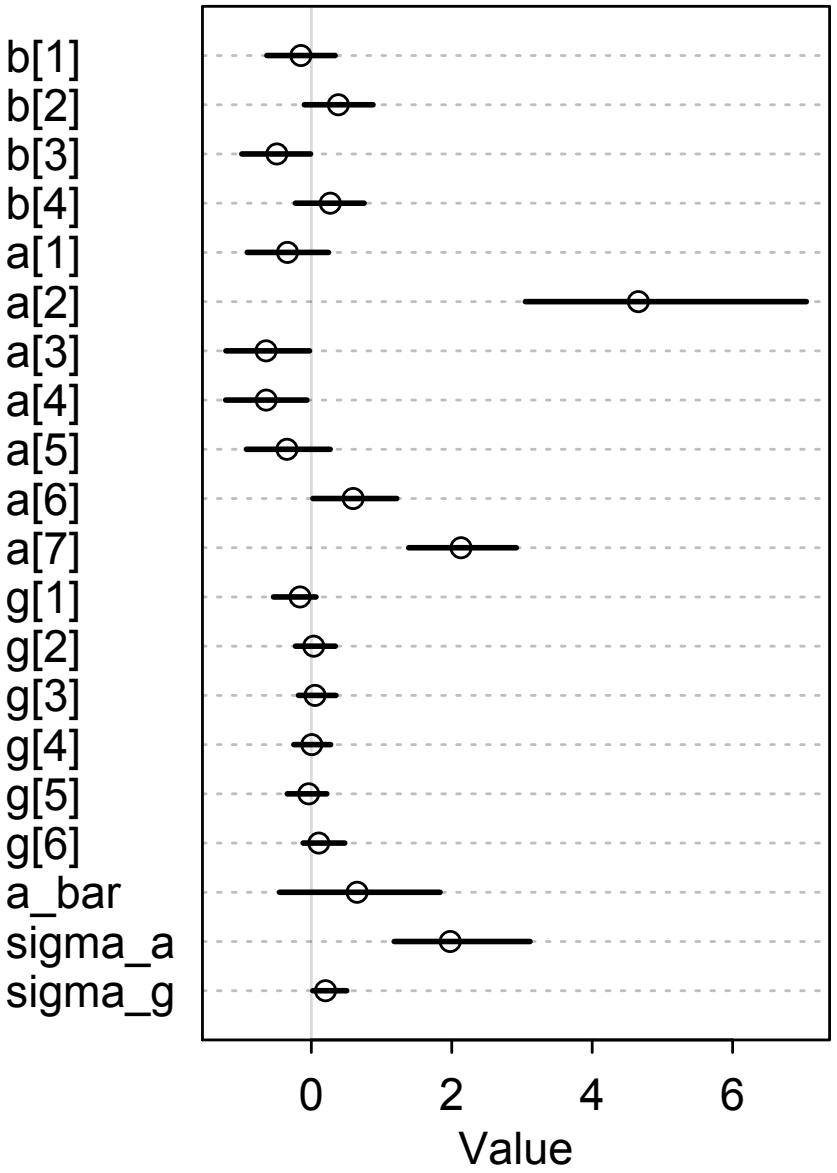
Multilevel chimpanzees

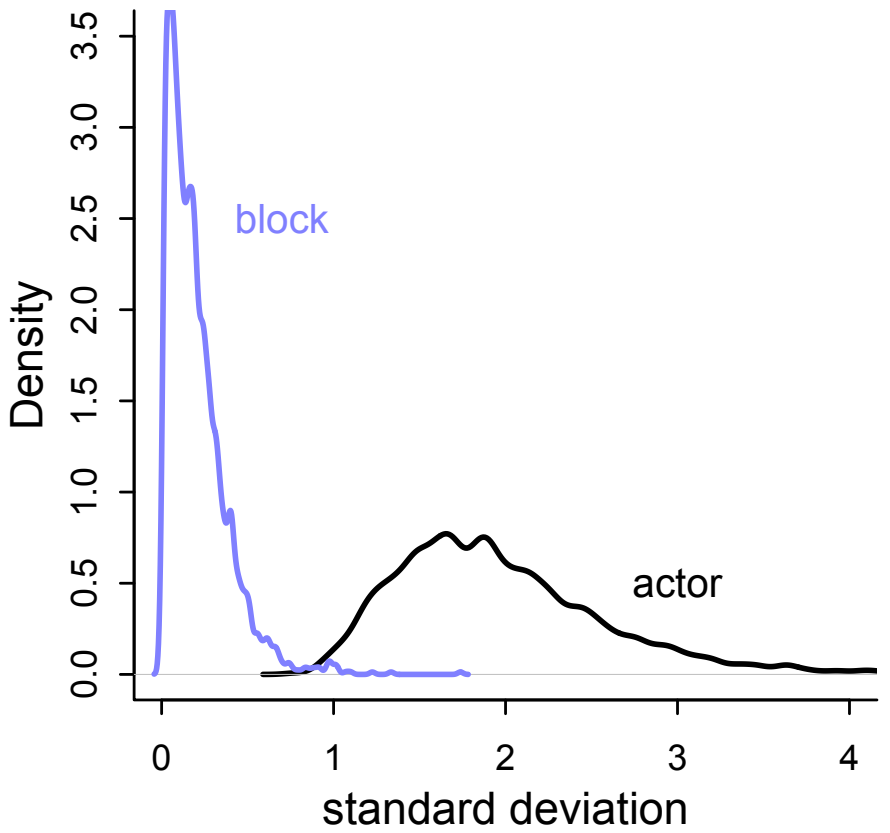
```
m13.4 <- ulam(
    alist(
        pulled_left ~ dbinom( 1 , p ) ,
        logit(p) <- a[actor] + g[block_id] + b[treatment] ,
        b[treatment] ~ dnorm( 0 , 0.5 ),

        # adaptive priors
        a[actor] ~ dnorm( a_bar , sigma_a ),
        g[block_id] ~ dnorm( 0 , sigma_g ),

        # hyper-priors
        a_bar ~ dnorm( 0 , 1.5 ),
        sigma_a ~ dexp(1),
        sigma_g ~ dexp(1)
    ) , data=dat_list , chains=4 , cores=4 , log_lik=TRUE )
```

Cross-classified chimpanzees





Cross-classified chimpanzees

- Incorporating block: no benefits; little cost

R code 13.23

```
set.seed(14)
m13.5 <- ulam(
    alist(
        pulled_left ~ dbinom( 1 , p ) ,
        logit(p) <- a[actor] + b[treatment] ,
        b[treatment] ~ dnorm( 0 , 0.5 ),
        a[actor] ~ dnorm( a_bar , sigma_a ),
        a_bar ~ dnorm( 0 , 1.5 ),
        sigma_a ~ dexp(1)
    ) , data=dat_list , chains=4 , cores=4 , log_lik=TRUE )
```

R code 13.24

```
compare( m13.4 , m13.5 )
```

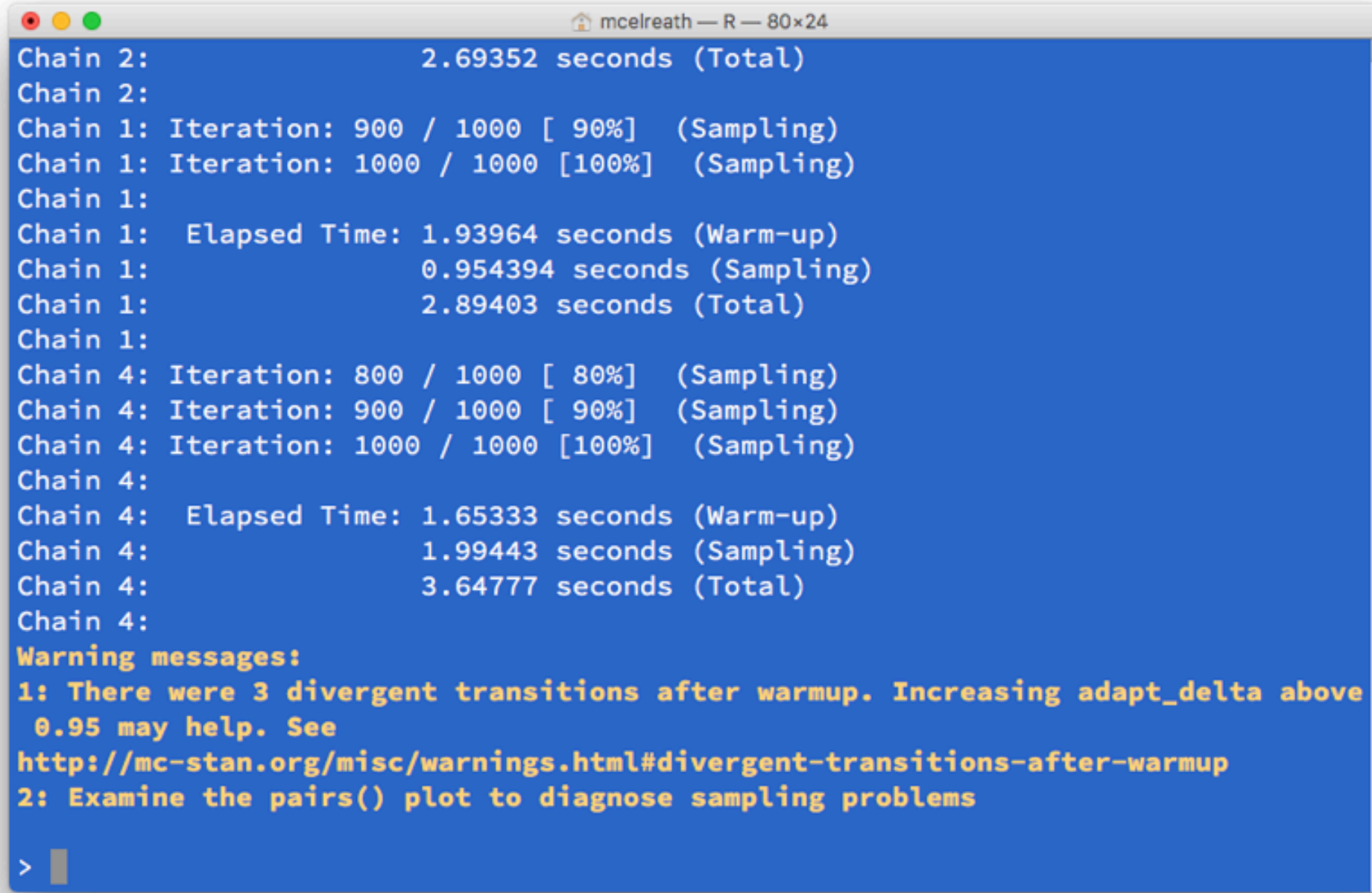
```
       WAIC pWAIC dWAIC weight    SE  dSE
m13.5 531.0   8.5     0   0.73 19.23   NA
m13.4 532.9  10.9     2   0.27 19.39 1.64
```

Everything is random

- "Random effects" have many definitions
- One common & bad definition:
  - Random effects are not fixed by the experiment
- Does not matter—we want pooling for estimation benefits
- Consider e.g. treatments

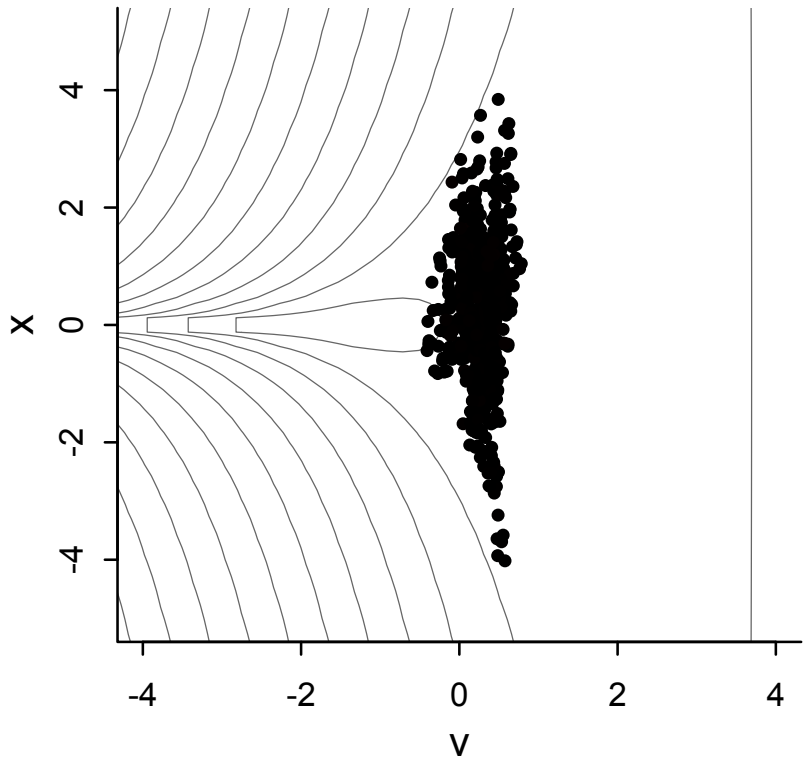
R code 13.25

```
set.seed(15)
m13.6 <- ulam(
    alist(
        pulled_left ~ dbinom( 1 , p ) ,
        logit(p) <- a[actor] + g[block_id] + b[treatment] ,
        b[treatment] ~ dnorm( 0 , sigma_b ),
        a[actor] ~ dnorm( a_bar , sigma_a ),
        g[block_id] ~ dnorm( 0 , sigma_g ),
        a_bar ~ dnorm( 0 , 1.5 ),
        sigma_a ~ dexp(1),
        sigma_g ~ dexp(1),
        sigma_b ~ dexp(1)
```


mcelreath — R — 80×24
Chain 2:   2.69352 seconds (Total)
Chain 2:
Chain 1: Iteration: 900 / 1000 [90%] (Sampling)
Chain 1: Iteration: 1000 / 1000 [100%] (Sampling)
Chain 1:
Chain 1: Elapsed Time: 1.93964 seconds (Warm-up)
Chain 1:   0.954394 seconds (Sampling)
Chain 1:   2.89403 seconds (Total)
Chain 1:
Chain 4: Iteration: 800 / 1000 [80%] (Sampling)
Chain 4: Iteration: 900 / 1000 [90%] (Sampling)
Chain 4: Iteration: 1000 / 1000 [100%] (Sampling)
Chain 4:
Chain 4: Elapsed Time: 1.65333 seconds (Warm-up)
Chain 4:   1.99443 seconds (Sampling)
Chain 4:   3.64777 seconds (Total)
Chain 4:
Warning messages:
1: There were 3 divergent transitions after warmup. Increasing adapt_delta above
 0.95 may help. See
http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
2: Examine the pairs() plot to diagnose sampling problems
>

Divergent transitions

- Two basic strategies:
- (1) Increase Stan's adapt_delta control parameter => better step size adaptation + slower exploration
- (2) Re-parameterize!



Re-parameterize!

- Most any statistical model can be expressed in several mathematically identical ways

$$\alpha \sim \text{Normal}(\mu, \sigma)$$ *Centered*

$$\alpha = \mu + \beta$$
$$\beta \sim \text{Normal}(0, \sigma)$$

$$\alpha = \mu + z\sigma$$
$$z \sim \text{Normal}(0, 1)$$ *Non-centered*

Re-parameterize!

- Why would this madness help with sampling?
- HMC sees a different geometry!

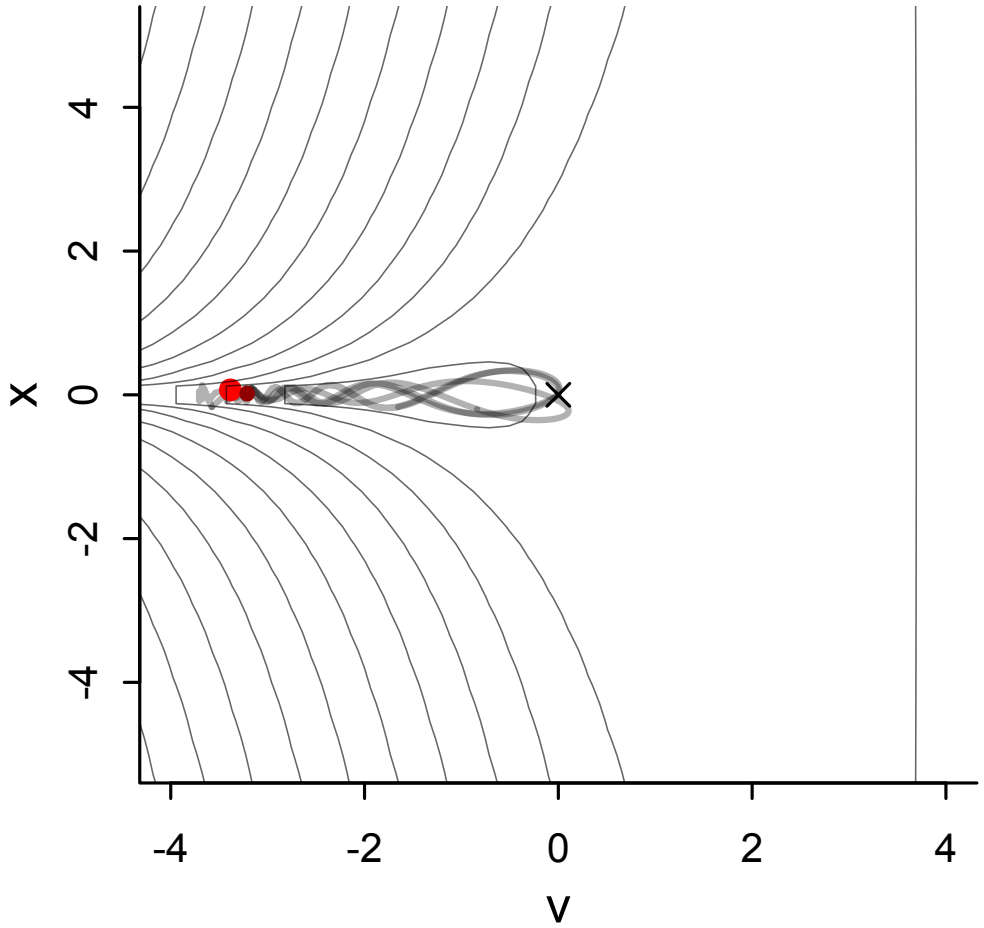
Centered

$$v \sim \text{Normal}(0, 3)$$
$$x \sim \text{Normal}(0, \exp(v))$$

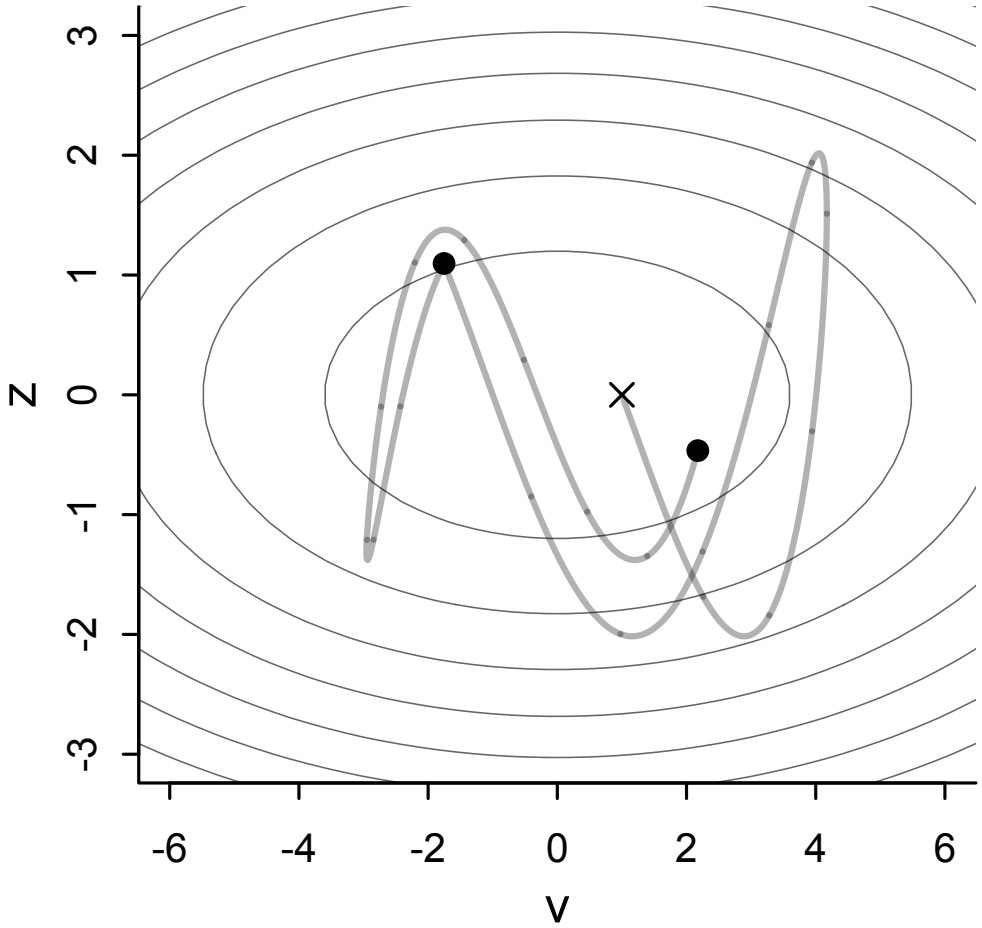
Non-centered

$$v \sim \text{Normal}(0, 3)$$
$$x = z \exp(v)$$
$$z \sim \text{Normal}(0, 1)$$

$$v \sim \text{Normal}(0, 3)$$
$$x \sim \text{Normal}(0, \exp(v))$$



$$v \sim \text{Normal}(0, 3)$$
$$x = z \exp(v)$$
$$z \sim \text{Normal}(0, 1)$$



Posterior predictions

- Predictions more subtle: Same clusters or new clusters?
- Same clusters: proceed as usual
- New clusters: should average over distribution of varying effects
- In this case:
  - Same clusters: Predictions for these chimpanzees
  - New clusters: Prediction for a new chimpanzee or rather for population of chimpanzees

Same clusters, new clusters

- Same actors:
  - Really same as before: varying effects are just parameters; you know the model; push samples back through the model
  - link() and sim() obey this rule
- New actors (counterfactual):
  - which actor (cluster) to use for counterfactual predictions?
    - average actor
    - marginal of actor
    - show sample of actors from posterior

Average actor

- "average actor" means actor with population average intercept, "alpha"
- Strategy:
  - replace varying intercept samples with zeros => all actors have average intercept now
  - compute predictions as usual

```
# replace varying intercept samples with zeros
# 1000 samples by 7 actors
a_actor_zeros <- matrix(0,1000,7)
```

R code 12.32

```
# fire up link
# note use of replace list
link.m12.4 <- link( m12.4 , n=1000 , data=d.pred ,
    replace=list(a_actor=a_actor_zeros) )
```

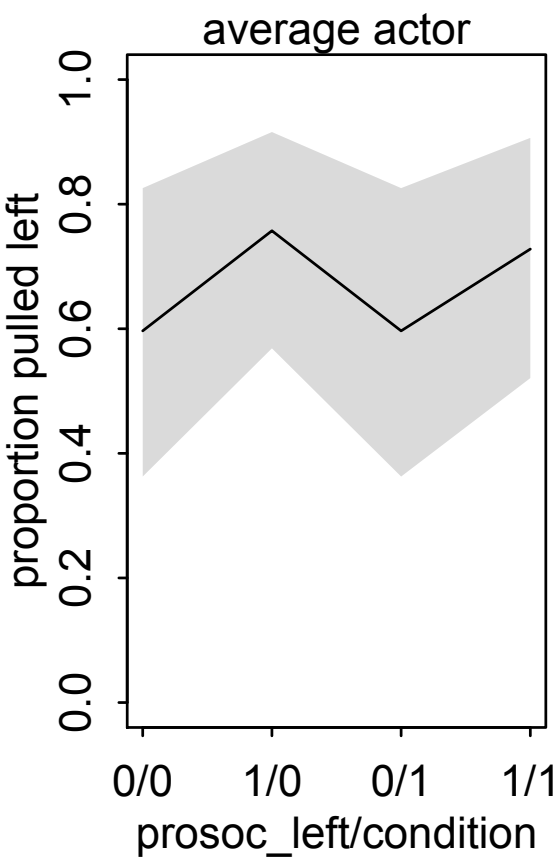
R code 12.33

R code 12.32

```
# replace varying intercept samples with zeros
# 1000 samples by 7 actors
a_actor_zeros <- matrix(0,1000,7)
```

R code 12.33

```
# fire up link
# note use of replace list
link.m12.4 <- link( m12.4 , n=1000 , data=d.pred ,
    replace=list(a_actor=a_actor_zeros) )
```



Marginal of actor

- "Marginal of" means "averaging over variation in actors" => shows variation arising from variation across actors

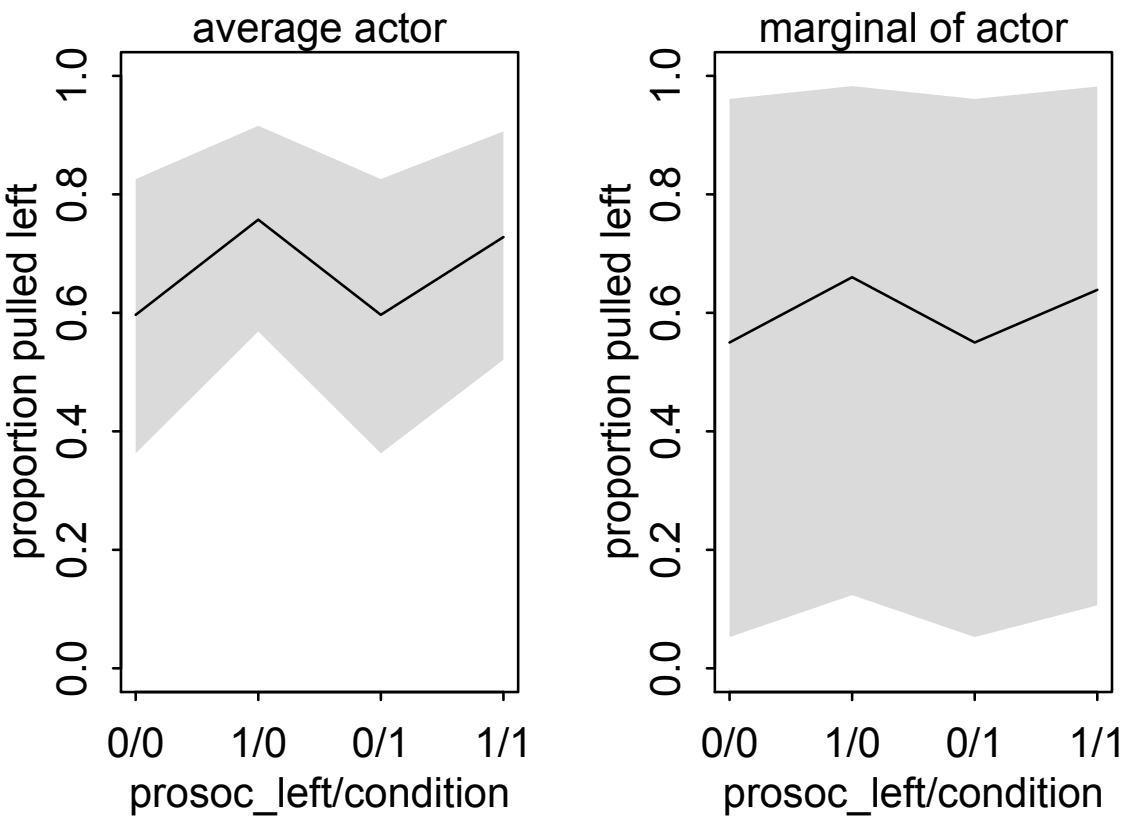
$$\alpha_{\text{ACTOR}} \sim \text{Normal}(0, \sigma_{\text{ACTOR}})$$

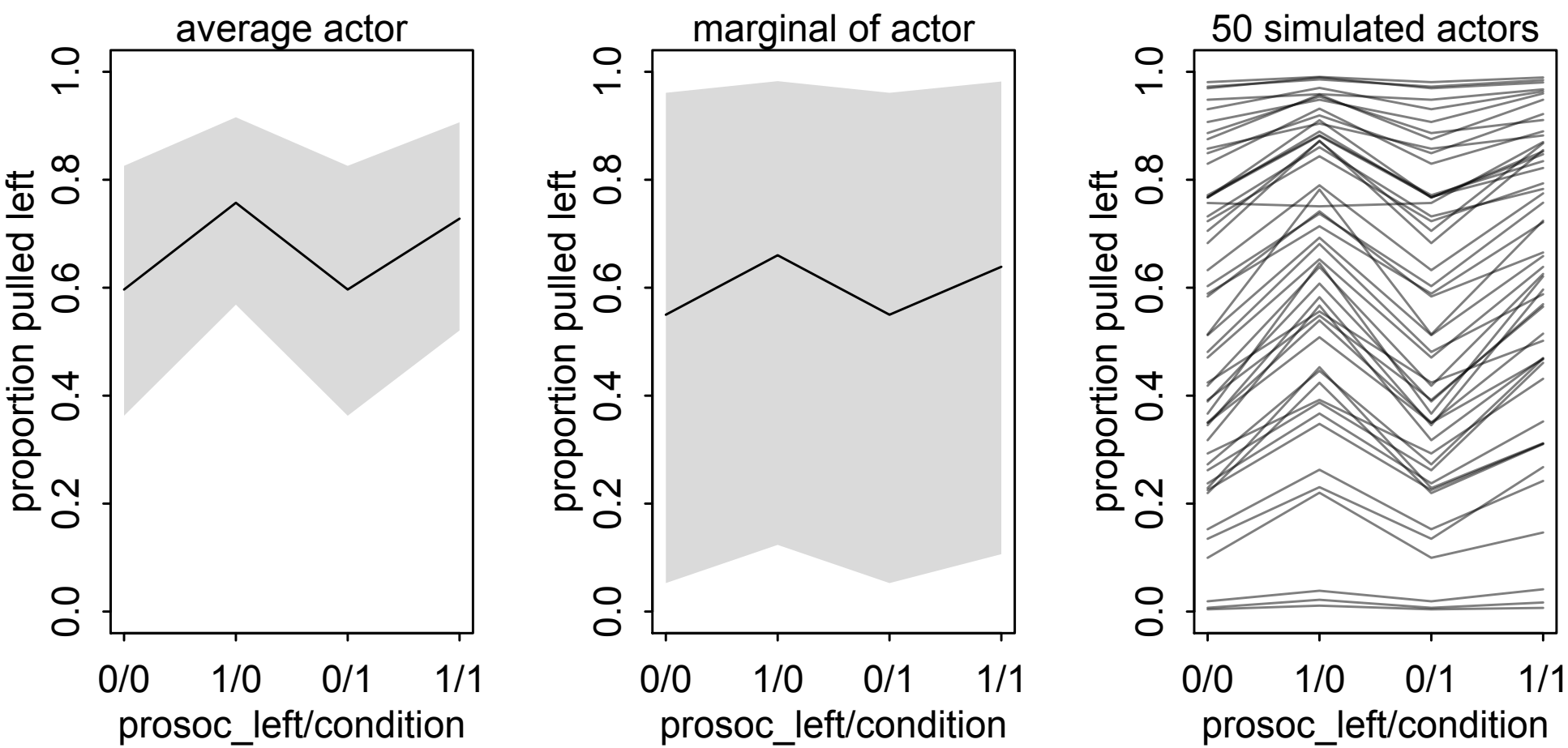
- Strategy:
  - Extract samples for sigma_actor
  - Simulate new varying intercepts
  - Use simulated intercepts to simulate predictions

R code 12.34

```
# replace varying intercept samples with simulations
post <- extract.samples(m12.4)
a_actor_sims <- rnorm(7000,0,post$sigma_actor)
a_actor_sims <- matrix(a_actor_sims,1000,7)

# fire up link
# note use of replace list
link.m12.4 <- link( m12.4 , n=1000 , data=d.pred ,
    replace=list(a_actor=a_actor_sims) )
```




average actor
marginal of actor
50 simulated actors
proportion pulled left
0.0
0.2
0.4
0.6
0.8
1.0
0/0
1/0
0/1
1/1
prosoc_left/condition