# Chimpanzees, part 2

*Stat 341 — Spring 2017*

*April, 2017*

### The Data

```
library(rethinking)
data(chimpanzees)
Chimps <- chimpanzees %>%
  mutate(
    recipient = ifelse(is.na(recipient), -1, recipient),  # to avoid missing data
    combo = paste0(prosoc_left, "/", condition))          # useful for plotting
Chimps %>% sample(3)  # 3 random rows
```

```
##     actor recipient condition block trial prosoc_left chose_prosoc pulled_left combo orig.id
## 225     4        -1         0     2    17           0            0           1   0/0     225
## 365     6        -1         0     1    10           0            0           1   0/0     365
## 96      2        -1         0     4    47           1            1           1   1/0      96
```

### Model with main effects, interaction, and varying intercepts

Note: This is not the same as model `m12.4` in the text.

```
m12.4a <- map2stan(
  alist(
    pulled_left ~ dbinom(1, p),
    logit(p) <- a + a_actor[actor] +
      bp * prosoc_left + bc * condition +  bpc * prosoc_left * condition,
    a_actor[actor] ~ dnorm(0, sigma_actor),
    c(a, bp, bc, bpc) ~ dnorm(0, 10),    # nifty short cut!
    sigma_actor ~ dcauchy(0, 1)
  ),
  data = Chimps,
  warmup = 1000, iter = 5000, chains = 4, cores = 3, refresh = 0
)
```

1. Describe how to modify the code above in a way that removes `a` from the second line of the formula list but still fits the "same model".
2. In what sense are the two models "the same"? In what ways are they different?

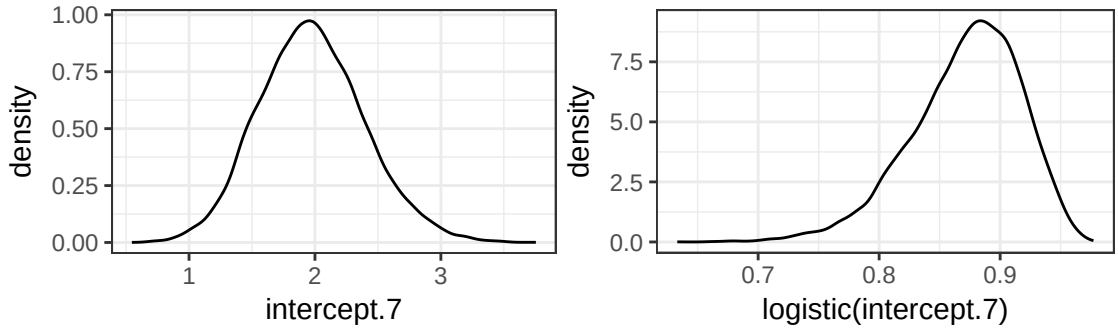
**Posterior samples**

```
m12.4a_post <- extract.samples(m12.4a)
glimpse(m12.4a_post)
```

```
## List of 6
##  $ a_actor    : num [1:16000, 1:7] -0.9048 -1.0586 0.3696 -0.5498 0.0517 ...
##  $ a          : num [1:16000(1d)] 0.1379 0.3277 -0.8717 0.0158 0.0439 ...
##  $ bp         : num [1:16000(1d)] 0.647 0.916 0.688 0.335 0.329 ...
##  $ bc         : num [1:16000(1d)] -0.505 -0.287 -0.319 -0.382 -0.658 ...
##  $ bpc        : num [1:16000(1d)] 0.5761 0.0208 0.3456 0.4342 1.0619 ...
##  $ sigma_actor: num [1:16000(1d)] 1.69 1.54 2.39 1.65 4.14 ...
```

3. How does the result of `extract.samples()` change now that we have a multi-level model?
4. In a varying intercpets model, each actor has its own intercept. In this model, what does the intercept for actor 7 represent?
5. In a Bayesian model, there is a posterior distribution for this intercept. Write a little R code that plots the posterior distribution of the intercept for actor 7.

```
gf_dens(~intercept.7, data = m12.4a_postD)
gf_dens(~ logistic(intercept.7), data = m12.4a_postD)
```




## A model with multiple clusters

Again, this is not quite the same as in the book. I've added the other main effect.

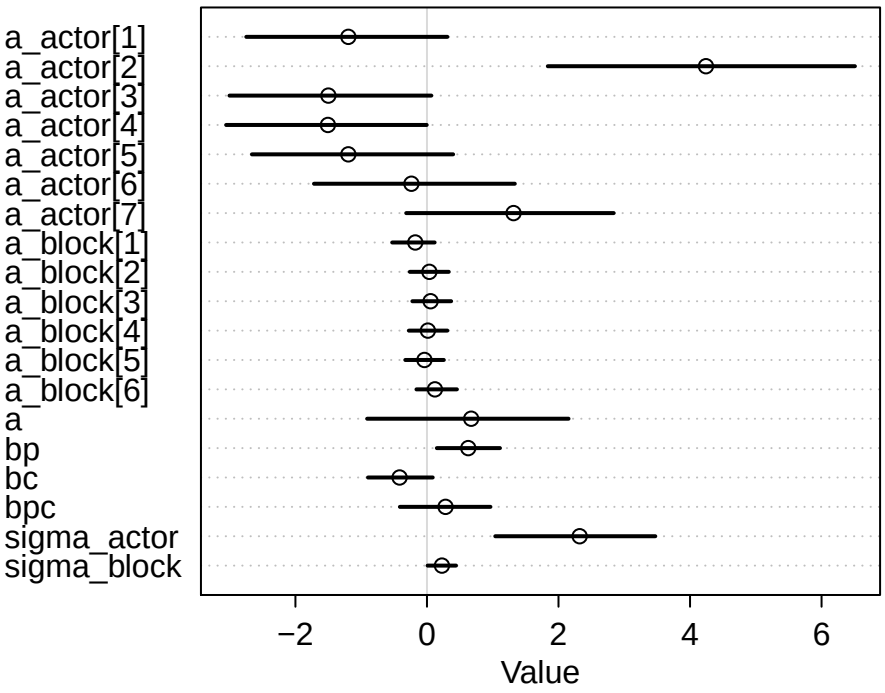
```
# prep data
Chimps <-
  Chimps %>% rename(block_id = block)   # name 'block' is reserved by Stan

m12.5a <- map2stan(
  alist(
    pulled_left ~ dbinom(1, p),
    logit(p) <- a + a_actor[actor] + a_block[block_id] +
      bp * prosoc_left + bc * condition + bpc * condition * prosoc_left,
    a_actor[actor] ~ dnorm(0, sigma_actor),
    a_block[block_id] ~ dnorm(0, sigma_block),
    c(a, bp, bc, bpc) ~ dnorm(0, 10),             # short cut again
    c(sigma_actor, sigma_block) ~ dcauchy(0, 1)   # and again
  ),
  data = Chimps, warmup = 1000, iter = 6000, chains = 4, cores = 3, refresh = 0
)
```

6. Explain how `m12.5a` differs from `m12.4a`.

```
coef(m12.5a)
```

```
##  a_actor[1]  a_actor[2]  a_actor[3]  a_actor[4]  a_actor[5]  a_actor[6]  a_actor[7]
## -1.19545192  4.24205314 -1.49905491 -1.50601607 -1.19542462 -0.23643461  1.31639119
##  a_block[1]  a_block[2]  a_block[3]  a_block[4]  a_block[5]  a_block[6]           a
## -0.17768577  0.03651059  0.05526852  0.01376358 -0.03854017  0.12089453  0.67240324
##          bp          bc         bpc sigma_actor sigma_block
##  0.62792839 -0.41643357  0.28225948  2.32175193  0.22768948
```

```
# precis(m12.5a, depth = 2)     # saving some paper
plot(precis(m12.5a, depth = 2)) # just show the plot
```




7. What do you noitice when you compare actors to blocks?

8. How do the comparisions below fit into this story?

```
compare(m12.4a, m12.5a)@output %>%
  mutate(
    logLik = c(logLik(m12.4a), m12.5a = logLik(m12.5a)),
    dlogLik = logLik - logLik(m12.4a))
```

```
##       WAIC    pWAIC    dWAIC    weight       SE      dSE    logLik  dlogLik
## 1 531.7753  9.21556 0.000000 0.6315959 19.76806       NA -256.1056 0.000000
## 2 532.8534 11.43523 1.078138 0.3684041 20.00556 1.783302 -254.3809 1.724775
```

### Posterior samples

Here's a quick trick for turning the list produced by `extract.samples()` into a data frame you can use for plotting. Noice the way the names change.

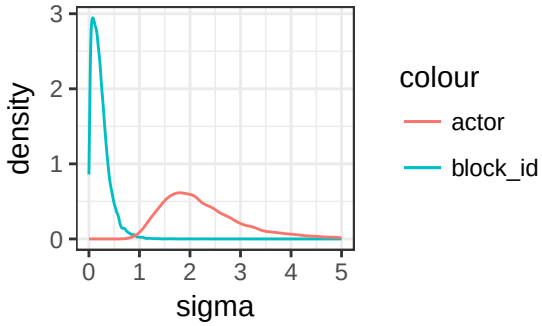
```
m12.5a_post <- extract.samples(m12.5a) %>% data.frame()
names(m12.5a_post)
```

```
##  [1] "a_actor.1"   "a_actor.2"   "a_actor.3"   "a_actor.4"   "a_actor.5"   "a_actor.6"
```

```
##  [7] "a_actor.7"   "a_block.1"   "a_block.2"   "a_block.3"   "a_block.4"   "a_block.5"
## [13] "a_block.6"   "a"           "bp"          "bc"          "bpc"         "sigma_actor"
## [19] "sigma_block"
```

```
gf_dens(~sigma_block + color::"block_id", data = m12.5a_post) %>%
  gf_dens(~sigma_actor + color::"actor", data = m12.5a_post) %>%
  gf_labs(x = "sigma") %>%
  gf_lims(x = c(0,5))
```

```
## Warning: Removed 445 rows containing non-finite values (stat_density).
```




**Converting to a "long" format.**

If we convert our posterior samples from a "wide" format (one row per sample) to a "long" format (one row for each sample of each parameter, we can plot posterior distributions for multiple parameters even more easily.

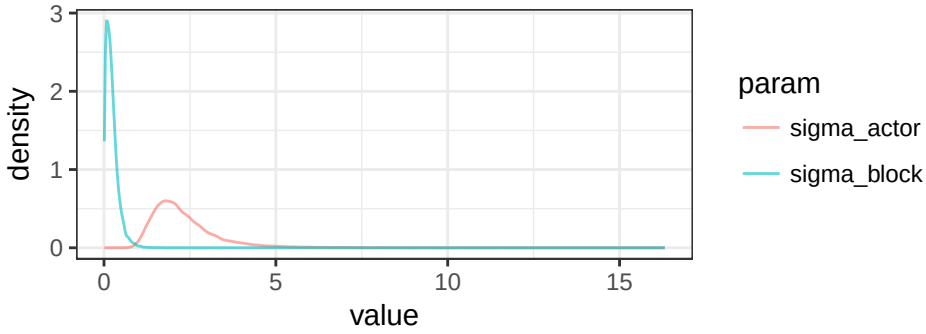
```
m12.5a_post_long <-  m12.5a_post %>%
          tidyr::gather(param, value)
```

```
## Warning: attributes are not identical across measure variables; they will be dropped
```

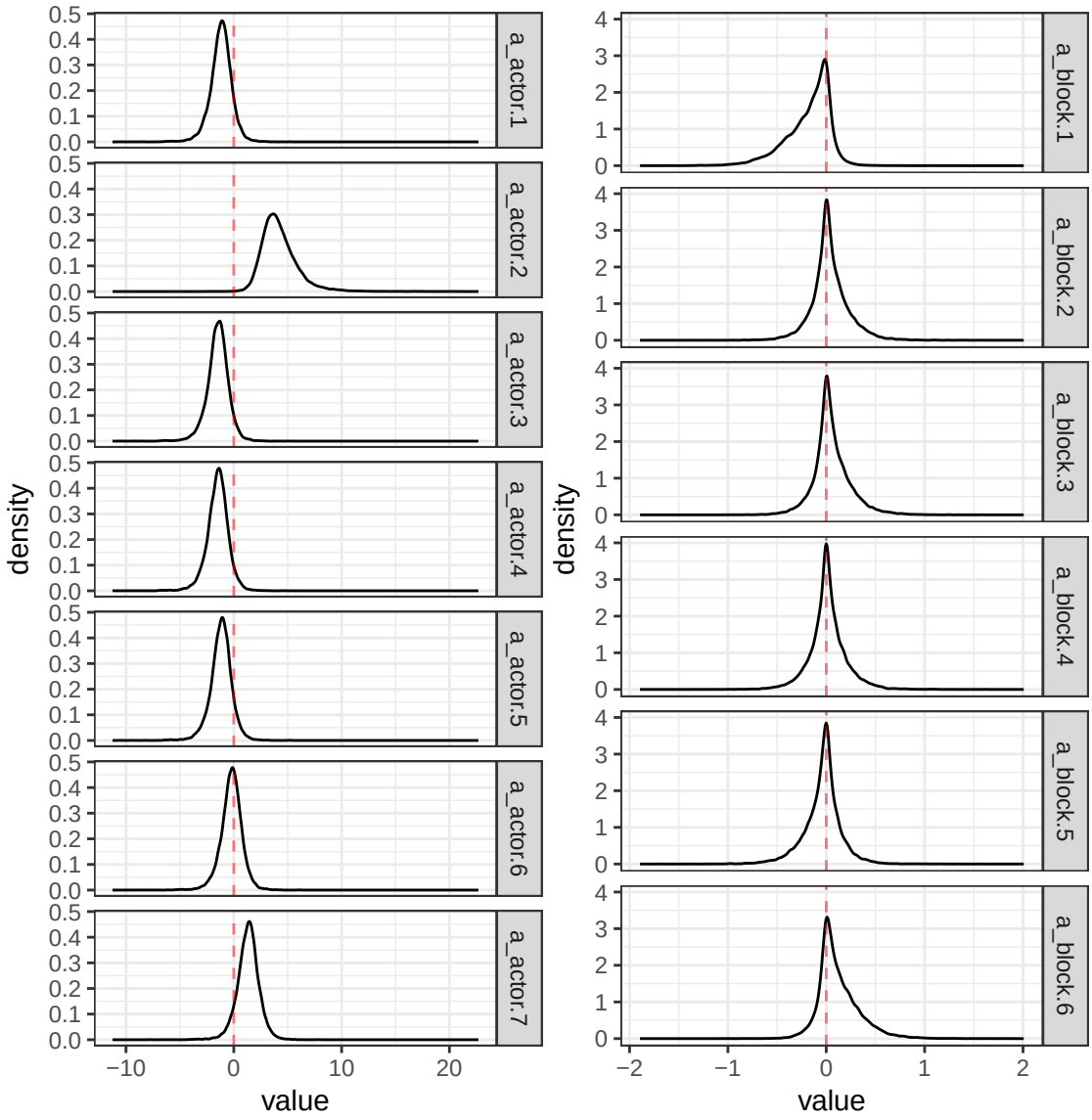
```
m12.5a_post_long %>% head(3)
```

```
##       param      value
## 1 a_actor.1 -0.7378205
## 2 a_actor.1 -0.5102513
## 3 a_actor.1  0.8929754
```

```
gf_dens(~ value + color::param, alpha = 0.6,
        data = m12.5a_post_long %>% filter(grepl("sigma", param)))
```



```
gf_dens(~ value,
        data = m12.5a_post_long %>% filter(grepl("a_actor\\.", param))) %>%
  gf_vline(xintercept = 0, color = "red", alpha = 0.5, linetype = "dashed") %>%
  gf_facet_grid( param ~ .)
gf_dens(~ value,
        data = m12.5a_post_long %>% filter(grepl("a_block\\.", param))) %>%
  gf_vline(xintercept = 0, color = "red", alpha = 0.5, linetype = "dashed") %>%
  gf_facet_grid( param ~ .)
```



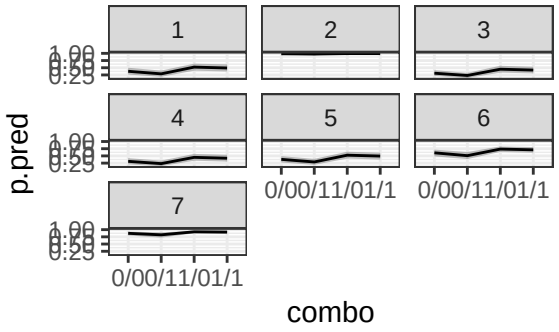
**R Code 12.27**

```
m12.4a_pred <-
  expand.grid(
    prosoc_left = 0:1,
    condition = 0:1,
    actor = 1:7) %>%
  mutate(
    combo = paste0(prosoc_left, "/", condition)
  )

link.m12.4a <- link(m12.4a, data = m12.4a_pred)
```

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

```
m12.4a_pred <- m12.4a_pred %>%
  mutate(
    p.pred = apply(link.m12.4a, 2, mean),
    p.link.lo = apply(link.m12.4a, 2, PI)[1,],
    p.link.hi = apply(link.m12.4a, 2, PI)[2,]
  )
```

```
gf_ribbon(p.link.lo + p.link.hi ~ combo + group::"1", data = m12.4a_pred) %>%
  gf_line(p.pred ~ combo + group::"1", data = m12.4a_pred) %>%
  gf_facet_wrap( ~ actor)
```




**R Code 12.28**

```
m12.4a_post <- extract.samples(m12.4a)
m12.4a_postD <- m12.4a_post %>% data.frame()
str(m12.4a_post)
```

```
## List of 6
##  $ a_actor    : num [1:16000, 1:7] -0.9048 -1.0586 0.3696 -0.5498 0.0517 ...
##  $ a          : num [1:16000(1d)] 0.1379 0.3277 -0.8717 0.0158 0.0439 ...
##  $ bp         : num [1:16000(1d)] 0.647 0.916 0.688 0.335 0.329 ...
```

```
##  $ bc         : num [1:16000(1d)] -0.505 -0.287 -0.319 -0.382 -0.658 ...
##  $ bpc        : num [1:16000(1d)] 0.5761 0.0208 0.3456 0.4342 1.0619 ...
##  $ sigma_actor: num [1:16000(1d)] 1.69 1.54 2.39 1.65 4.14 ...
```

```
str(m12.4a_postD)
```

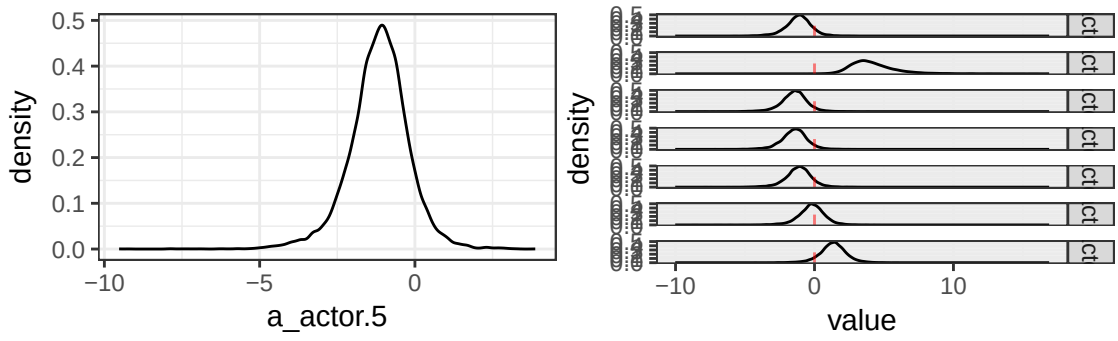
```
## 'data.frame':    16000 obs. of  12 variables:
##  $ a_actor.1  : num  -0.9048 -1.0586 0.3696 -0.5498 0.0517 ...
##  $ a_actor.2  : num  3.37 3.66 5.89 4.36 6.57 ...
##  $ a_actor.3  : num  -1.151 -1.541 0.176 -0.448 -1.13 ...
##  $ a_actor.4  : num  -1.311 -1.177 0.237 -0.617 -0.942 ...
##  $ a_actor.5  : num  -0.8478 -0.6893 0.0311 -0.2395 -0.5979 ...
##  $ a_actor.6  : num  0.0258 -0.1118 1.585 0.7382 0.5983 ...
##  $ a_actor.7  : num  2.2 1.55 2.63 1.98 2.41 ...
##  $ a          : num [1:16000(1d)] 0.1379 0.3277 -0.8717 0.0158 0.0439 ...
##  $ bp         : num [1:16000(1d)] 0.647 0.916 0.688 0.335 0.329 ...
##  $ bc         : num [1:16000(1d)] -0.505 -0.287 -0.319 -0.382 -0.658 ...
##  $ bpc        : num [1:16000(1d)] 0.5761 0.0208 0.3456 0.4342 1.0619 ...
##  $ sigma_actor: num [1:16000(1d)] 1.69 1.54 2.39 1.65 4.14 ...
```

**R Code 12.29**

```
gf_dens( ~ a_actor.5, data = m12.4a_postD)
m12.4a_post_long <-
  m12.4a_postD %>%
  tidyr::gather(param, value)
```

```
## Warning: attributes are not identical across measure variables; they will be dropped
```

```
gf_dens( ~ value, data = m12.4a_post_long %>% filter(grepl("actor\\.", param))) %>%
  gf_vline(xintercept = 0, color = "red", alpha = 0.5, linetype = "dashed") %>%
  gf_facet_grid( param ~ .)
```




### Creating the link function manually

For some models, it is necessary to create posterior values manually rather than with `link()` or `sim()`. Although this is not required here, we illustrate the manual approach by way of comparison.

**R Code 12.30**

```
p.link <- function(prosoc_left, condition, actor, post = m12.4a_post) {
  logodds <-
    with(post,
```

```
        a + a_actor[, actor] + (bp + bpc * condition) * prosoc_left)
  return(logistic(logodds))
}
str(p.link(0, 0, 2))
```

```
##  num [1:16000(1d)] 0.971 0.982 0.993 0.988 0.999 ...
```

**R Code 12.31**

```
G <- expand.grid(
  prosoc_left = 0:1,
  condition = 0:1,
  actor = 1:7) %>%
  mutate(
    combo = paste0(prosoc_left, "/", condition)
  )

manual_link <-
  with(G,    # saves having to type G$ repeatedly
       mapply(p.link, prosoc_left, condition, actor)
  )
# Note: a column here for each row in G
str(manual_link)
```

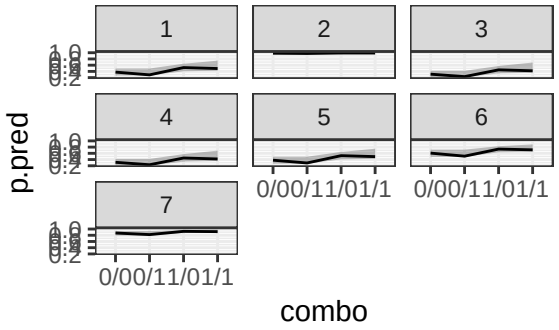
```
##  num [1:16000, 1:28] 0.317 0.325 0.377 0.37 0.524 ...
```

```
dim(G)
```

```
## [1] 28  4
```

```
manual12.4_pred <-
  G %>%
  mutate(
    p.pred = apply(manual_link, 2, mean),
    p.link.lo = apply(manual_link, 2, PI)[1,],
    p.link.hi = apply(manual_link, 2, PI)[2,]
  )
```

```
# replication of plot from above using home spun link
gf_ribbon(p.link.lo + p.link.hi ~ combo + group::"1", data = manual12.4_pred) %>%
  gf_line(p.pred ~ combo + group::"1", data = m12.4a_pred) %>%
  gf_facet_wrap( ~ actor)
```




**R Code 12.32**

```
# don't need multiple actors this time
G <- expand.grid(
  prosoc_left = 0:1,
  condition = 0:1,
  actor = 1) %>%
  mutate(
    combo = paste0(prosoc_left, "/", condition)
  )
```

**R Code 12.33**

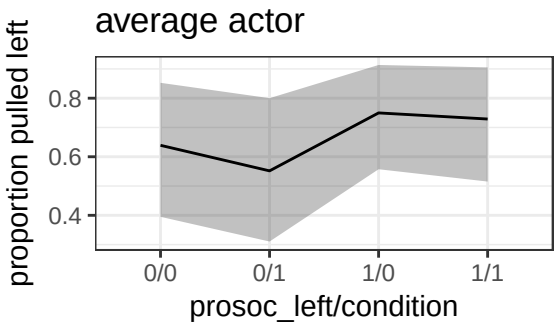
```
# replace varying intercept samples with zeros
# 1000 samples by 7 actors
a_actor_zeros <- matrix(0, 1000, 7)
```

**R Code 12.34**

```
# note use of replace list
m12.4a_link2 <-
  link(
    m12.4a,
    n = 1000,
    data = G,
    replace = list(a_actor = a_actor_zeros)
  )
```

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

```
m12.4a_pred2 <-
  G %>%
  mutate(
    p.pred = apply(m12.4a_link2, 2, mean),
    p.link.lo = apply(m12.4a_link2, 2, PI, prob = 0.8)[1,],
    p.link.hi = apply(m12.4a_link2, 2, PI, prob = 0.8)[2,]
  )


gf_ribbon(p.link.lo + p.link.hi ~ combo + group::"1", data = m12.4a_pred2) %>%
  gf_line(p.pred ~ combo + group::"doesn't matter what this is", data = m12.4a_pred2) %>%
  gf_labs(x = "prosoc_left/condition", y = "proportion pulled left", title = "average actor")
```




**R Code 12.35**

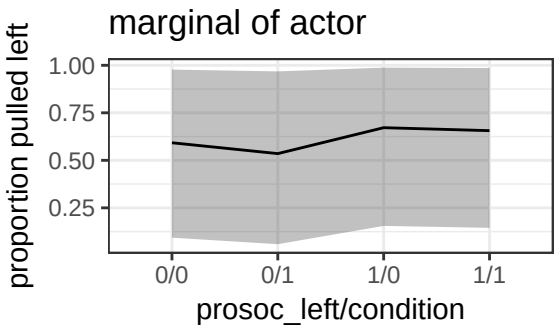
```
# replace varying intercept samples with simulations
m12.4a_post <- extract.samples(m12.4a)
a_actor_sims <-
  rnorm(7000, 0, m12.4a_post$sigma_actor) %>%
  matrix(1000, 7)    # reshape into a 1000 x 7 matrix
```

**R Code 12.36**

```
m12.4a_link3 <-
  link(
    m12.4a,
    n = 1000,
    data = G,
    replace = list(a_actor = a_actor_sims)
  )
```

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

```
m12.4a_pred3 <-
  G %>%
  mutate(
    p.pred = apply(m12.4a_link3, 2, mean),
    p.link.lo = apply(m12.4a_link3, 2, PI, prob = 0.8)[1,],
    p.link.hi = apply(m12.4a_link3, 2, PI, prob = 0.8)[2,]
  )


gf_ribbon(p.link.lo + p.link.hi ~ combo + group::"1", data = m12.4a_pred3) %>%
  gf_line(p.pred ~ combo + group::"doesn't matter what this is", data = m12.4a_pred3) %>%
  gf_labs(x = "prosoc_left/condition", y = "proportion pulled left", title = "marginal of actor")
```




**R Code 12.37**

```
m12.4a_post <- extract.samples(m12.4a) %>%
  as.data.frame() %>%
  mutate(
    sim_a_actor = rnorm(16000, 0, sigma_actor)
  )

Actors50 <-
  expand.grid(
    actor = 1:50,         # 50 simulated actors
    prosoc_left = 0:1,
    condition = 0:1) %>%
  mutate(
    combo = paste0(prosoc_left, "/", condition),
    logodds =
      m12.4a_post$a[actor] +
      m12.4a_post$sim_a_actor[actor] +
      (m12.4a_post$bp[actor] + m12.4a_post$bpc[actor] * condition) * prosoc_left,
    p = logistic(logodds)
  )
```

**R Code 12.38**

```
gf_line(p ~ combo + group::actor, data = Actors50, alpha = 0.3) %>%
  gf_labs(
    x = "prosoc_left/condition",
    y = "proportion pulled left",
    title = "50 simulated actors"
  )
```