Partial pooling test

Zane

2/14/23

Based on conversations with Amanda and Andreas, we decided to switch from trying to make a CI out of the individual fits to fitting a multilevel model using a Bayesian partial pooling approach. This script is my first test of this idea.

# Introduction

One of the main concepts for this paper was to fit linear regressions on titer increase vs. antigenic distance using the UGAFluVac data. This linear model acts like a summary of the information contained in the antigenic distance data, and can be used to quantify both the strength and breadth of the response. Going forward we adopt this terminology:

* **Strength** of the response: quantifies how much of an immune response is generated to the homologous strain.
* **Breadth** of the response: quantifies how much of an immune response is generated to antigenically distinct strains.
* **Overall magnitude of the response**: combines strength and breadth together into an idea of how much of an immune response was induced overall.

The basic idea of our framework for quantifying these things is that the AUC of the linear regression line estimates the overall magnitude of the response, and depends on the slope and intercept of the line. The slope captures information about the breadth of the response while the intercept captures information about the strength of the response. (See the trapezoids.md doc for a few brief thoughts I had about this.)

In the first round of tests, we used a simple linear regression model (a “complete pooling” model in bayesian terminology) as the estimator. ([Figure 1](#fig-thesis)).

|  |
| --- |
| Figure 1: From Amanda’s thesis: The titer increase to the H1N1 and H3N2 virus panels for the 2017 season of all individuals who received SD vaccination. The columns are separated by vaccine strain. The linear regression of the titerincrease with 95% confidence intervals is shown for each distance method. The distances were normalized by season. The raw data points had jitter applied with +/- 0.4 in the y-axis. Raw data points that fell outside of the y-axis bounds are not shown. |

However, we then realized that there was a lot of variation in trajectories between individuals. [Figure 2](#fig-indivLines) shows the regression line for each individual. After seeing this, one of our main concerns was that there appears to be a significant amount of heterogeneity in the data. Therefore, it would be better to estimate the variability in the fitted regression line from this “no pooling” model rather than from the complete pooling model.

|  |
| --- |
| Figure 2: From Amanda’s email: I’ve added the individual linear regression lines in the background. However, with the two colors/linetypes it’s very difficult to distinguish the main two lines for the groupings. This is an image that would go into Figure 1. Also it’s for MI/15 instead of CA/09 so there about 200 less individuals (400 less lines considering pre/post titer) present on these plots compared to what would be later for CA/09. Also, with the change in the HAI virus panel the domains of the linear regressions are different. Although it provides more transparency in that aspect it does add to the confusion. |

From this figure, we can see that there are significant amount of people who had a much smaller panel that didn’t include the most distant viruses. Therefore, points with moderate antigenic distances have much more leverage and systematically drive their slopes down. **We still need to investigate whether** **these points have similarly low values for patients with the full panel.** There are two explanations here from my perspective.

1. These people with steeper slopes have systematic differences that make their responses less broad.
2. Everyone has a dip in the middle, but for people with the entire range of historical viruses in their panel, these points have much lower leverage and the regression line smooths over this bump in the trajectory.

Anyways, we then ran into trouble trying to get some kind of average estimate (marginalized over individuals) from the no pooling estimate, but this is quite hard. [Figure 3](#fig-bounds) shows Amanda’s attempts at combining the average slope and intercept into boundaries. My proposal for this method was to take the empirical quantiles of the regression predictions over every x-value, but this got us into the weeds of the best way to compute this interval.

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | (a) I’ll include the bounding lines for the titer = slope\_upperSD \* distance + intercept\_upperSD (purple line) and opposite for lower bound (green line) since they include the other combinations (slope\_lwr + intercept\_upr (pink line). | | |  | | --- | | (b) The linear regression and the averaged Linear regression do not match. If the ranges of the data were the same it wouldn’t be a problem but the data that does not have as large of a range have a steeper slope and are pulling the averaged linear regression slope results to be steeper. This isn’t matched in the linear regression with all of the data since it doesn’t extrapolate for those points. | |

Figure 3: From emails.

So Andreas and I both agreed that one potential solution to this CI issue would be **partial pooling**: using a multilevel model to allow the individual slopes to borrow from the overall mean if they are low precision or extremely different, but allowing for much more individual variation. We could use a frequentist mixed-effects model (e.g. through lme4) for this, but we also decided it would be easier to switch to a Bayesian hierarchical model at this time.

# The model (math part)

For this example, we’ll only fit a model to the 2017 season and the H1N1-Michigan-2015 strain as a proof-of-concept. Then we can incorporate season and strain effects as well.

## No pooling

This model has no pooling, the slopes are estimated with each individual’s data and they do not share data between individuals.

In this model, indexes the individuals, and is an index variable denoting whether the measurement is pre- or post-vaccination. Note that this naive model does not implement a correlation between the pre- and post-vaccination parameters for a given individual. Basically this fits completely separate models for the pre and post vaccination titers, but they are bundled together into one neat model formula. This model makes the assumptions that individuals have the same variance within each time group.

## Adaptively pooled

But we also want to try an adaptively pooled model, because this is really the best way to find a compromise between the no pooling and the complete pooling estimates. **Probably need to work out what notation we want to use in our group.** **Not sure the notation I used here is ideal.**

## Complete pooling

# Model fitting (computational part)

First I’ll fit the models with quap from the rethinking package because this is an easy way to make sure I write down the model I want.

We’ll need to do a small amount of preparatory data cleaning.

clean\_data <- readr::read\_rds(  
 here::here("data", "processed", "distance\_data.rds")  
)  
dat <-  
 clean\_data |>  
 dplyr::filter(  
 vaccine\_fullname == "H1N1-Michigan-2015",  
 strain\_type == "H1N1",  
 # Just use year distance for now  
 method == "year"  
 ) |>  
 dplyr::mutate(  
 dplyr::across(tidyselect:::where(is.factor), forcats::fct\_drop)  
 )  
# Pivot the data so we can fit pre/post models at the same time  
pivoted <-  
 dat |>  
 dplyr::select(  
 uniq\_id, age, dose, strains\_fullname, prevactiter, postvactiter, distance  
 ) |>  
 tidyr::pivot\_longer(  
 cols = c(prevactiter, postvactiter),  
 names\_to = c("time"),  
 values\_to = c("titer")  
 ) |>  
 # Turn the categorical variables into integer indexes, this is required  
 # for index coding in stan  
 dplyr::mutate(  
 id = uniq\_id |>  
 factor() |>  
 forcats::fct\_inorder() |>  
 as.integer(),  
 t = time |>  
 factor(levels = c("prevactiter", "postvactiter")) |>  
 as.integer()  
 )  
# Split to pre and post data to avoid issues with Stan model  
pivoted\_pre <- dplyr::filter(pivoted, t == 1)  
pivoted\_post <- dplyr::filter(pivoted, t == 2)  
# Data list for Stan fitting  
d\_pre <-  
 list(  
 # Convert IDs to integer index  
 id = pivoted\_pre$id,  
 y = pivoted\_pre$titer,  
 x = pivoted\_pre$distance  
 )  
d\_post <-  
 list(  
 # Convert IDs to integer index  
 id = pivoted\_post$id,  
 y = pivoted\_post$titer,  
 x = pivoted\_post$distance  
 )

## No pooling

First we’ll fit the model without adaptive priors.

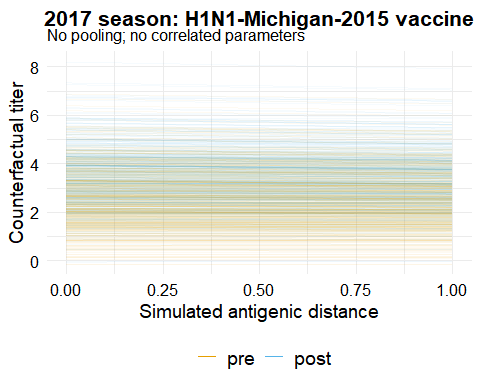
set.seed(12312)  
N <- 1000  
m1\_pre <-  
 rethinking::ulam(  
 alist(  
 y ~ dnorm(mu, s),  
 mu <- a[id] + b[id] \* x,  
 a[id] ~ dnorm(5, 5),  
 b[id] ~ dnorm(0, 5),  
 s ~ dexp(1)  
 ),  
 data = d\_pre,  
 chains = 4,  
 cores = 4,  
 iter = N  
 )  
m1\_post <-  
 rethinking::ulam(  
 alist(  
 y ~ dnorm(mu, s),  
 mu <- a[id] + b[id] \* x,  
 a[id] ~ dnorm(5, 5),  
 b[id] ~ dnorm(0, 5),  
 s ~ dexp(1)  
 ),  
 data = d\_post,  
 chains = 4,  
 cores = 4,  
 iter = N  
 )

Next we can use the posterior samples to estimate the counterfactual effect of “varying” the antigenic distance for each individual – based on the information we have, this is what we would expect their response to be for a given antigenic distance, assuming the model is correct.

# Simulate the counterfactual effects or whatever  
sim\_dat <-  
 tidyr::expand\_grid(  
 id = unique(d\_pre$id),  
 x = seq(0, 1, 0.01)  
 )  
# Force the garbage collector to run  
gc()  
pre <- list()  
pre$samples <- rethinking::link(m1\_pre, data = sim\_dat, n = N)  
pre$means <- colMeans(pre$samples)  
pre$PIs <- apply(pre$samples, 2, rethinking::PI)  
# Force the garbage collector to run  
gc()  
post <- list()  
post$samples <- rethinking::link(m1\_post, data = sim\_dat, n = N)  
post$means <- colMeans(post$samples)  
post$PIs <- apply(post$samples, 2, rethinking::PI)  
# Fcn to put this stuff into tibble  
clean\_list <- function(ls) {  
 tibble::tibble(  
 est = ls$means,  
 lwr = ls$PIs[1, ],  
 upr = ls$PIs[2, ]  
 )  
}  
# Add the simulations to the dataset -- this code looks silly but I couldn't  
# think of a better way  
preds <-  
 dplyr::bind\_cols(  
 dplyr::bind\_rows(  
 sim\_dat,  
 sim\_dat  
 ),  
 dplyr::bind\_rows(  
 "1" = clean\_list(pre),  
 "2" = clean\_list(post),  
 .id = "t"  
 )  
 )

This is a plot of the individual counterfactual simualtions.

library(ggplot2)  
preds |>  
 dplyr::mutate(  
 t = factor(t, levels = c("1", "2"), labels = c("pre", "post"))  
 ) |>  
 ggplot() +  
 aes(  
 x = x, y = est,  
 group = paste(id, t),  
 color = t  
 ) +  
 geom\_line(alpha = 0.05) +  
 labs(  
 x = "Simulated antigenic distance",  
 y = "Counterfactual titer",  
 color = NULL,  
 title = "2017 season: H1N1-Michigan-2015 vaccine",  
 subtitle = "No pooling; no correlated parameters"  
 ) +  
 scale\_color\_manual(values = c("#E69F00", "#56B4E9")) +  
 guides(color = guide\_legend(override.aes = list(alpha = 1))) +  
 zlib::theme\_ms()



Now we want to answer the question about the population, not about individuals. We can do that by constructing an estimate the marginalizes over the individual effects. The simplest way to do this (but representing the least amount of uncertainty) is by using the average of the estimated parameters, and constructing means and equal-tailed credible intervals from the samples of the means.

# Marginalization? Prediction for new AVERAGE individual  
# See rethinking pg 428  
pred\_indiv <- function(x, a, b) {  
 out <- a + b \* x  
 return(out)  
}  
# Marginalize pre  
pre\_posterior <- rethinking::extract.samples(m1\_pre)  
a\_bar <- rowMeans(pre\_posterior$a)  
b\_bar <- rowMeans(pre\_posterior$b)  
raw\_pre <- sapply(seq(0, 1, 0.01), function(x) pred\_indiv(x, a\_bar, b\_bar))  
pre\_mu <- colMeans(raw\_pre)  
pre\_PI <- apply(raw\_pre, 2, rethinking::PI)  
# Marginalize post  
post\_posterior <- rethinking::extract.samples(m1\_post)  
a\_bar <- rowMeans(post\_posterior$a)  
b\_bar <- rowMeans(post\_posterior$b)  
raw\_post <- sapply(seq(0, 1, 0.01), function(x) pred\_indiv(x, a\_bar, b\_bar))  
post\_mu <- colMeans(raw\_post)  
post\_PI <- apply(raw\_post, 2, rethinking::PI)  
# Make the data frame  
means\_dat <-  
 tibble::tibble(  
 x = c(seq(0, 1, 0.01), seq(0, 1, 0.01)),  
 est = c(pre\_mu, post\_mu),  
 lwr = c(pre\_PI[1, ], post\_PI[1, ]),  
 upr = c(pre\_PI[2, ], post\_PI[2, ]),  
 time = factor(  
 rep(c("pre", "post"), each = length(pre\_mu)),  
 levels = c("pre", "post")  
 )  
 )

p1 <-preds |>  
 dplyr::mutate(  
 t = factor(t, levels = c("1", "2"), labels = c("pre", "post"))  
 ) |>  
 ggplot() +  
 aes(  
 x = x, y = est,  
 group = paste(id, t),  
 color = t  
 ) +  
 geom\_line(alpha = 0.05) +  
 geom\_ribbon(  
 data = means\_dat,  
 aes(x = x, ymin = lwr, ymax = upr, y = est, fill = time),  
 alpha = 0.5,  
 inherit.aes = FALSE,  
 show.legend = FALSE  
 ) +  
 geom\_line(  
 data = means\_dat,  
 aes(x = x, y = est, color = time),  
 linewidth = 0.95, alpha = 1,  
 inherit.aes = FALSE  
 ) +  
 labs(  
 x = "\nSimulated antigenic distance",  
 y = "Counterfactual titer\n",  
 color = NULL,  
 subtitle = "No pooling"  
 ) +  
 scale\_color\_manual(values = rev(c("#E69F00", "#56B4E9"))) +  
 scale\_fill\_manual(values = rev(c("#E69F00", "#56B4E9"))) +  
 scale\_y\_continuous(  
 limits = c(-2.5, 9),  
 breaks = seq(-2, 8, 2)  
 ) +  
 coord\_cartesian(expand = FALSE) +  
 guides(color = guide\_legend(override.aes = list(alpha = 1))) +  
 zlib::theme\_ms() +  
 theme(  
 plot.title = element\_text(hjust = 0, margin = margin(0, 0, 5, 0))  
 )

## Adaptive

Next we’ll fit the models with adaptive priors.

set.seed(12312)  
N <- 1000  
m2\_pre <-  
 rethinking::ulam(  
 alist(  
 y ~ dnorm(mu, s),  
 mu <- a[id] + b[id] \* x,  
 a[id] ~ dnorm(abar, phi),  
 b[id] ~ dnorm(bbar, psi),  
 abar ~ dnorm(5, 5),  
 bbar ~ dnorm(0, 1),  
 phi ~ dexp(1),  
 psi ~ dexp(1),  
 s ~ dexp(1)  
 ),  
 data = d\_pre,  
 chains = 4,  
 cores = 4,  
 iter = N  
 )  
m2\_post <-  
 rethinking::ulam(  
 alist(  
 y ~ dnorm(mu, s),  
 mu <- a[id] + b[id] \* x,  
 a[id] ~ dnorm(abar, phi),  
 b[id] ~ dnorm(bbar, psi),  
 abar ~ dnorm(5, 5),  
 bbar ~ dnorm(0, 1),  
 phi ~ dexp(1),  
 psi ~ dexp(1),  
 s ~ dexp(1)  
 ),  
 data = d\_post,  
 chains = 4,  
 cores = 4,  
 iter = N  
 )

Get the simulations

# Force the garbage collector to run  
gc()

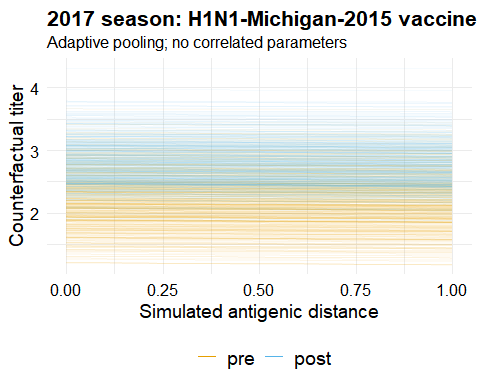
used (Mb) gc trigger (Mb) max used (Mb)  
Ncells 1900295 101.5 2834734 151.4 2834734 151.4  
Vcells 249530382 1903.8 777212660 5929.7 574363475 4382.1

pre <- list()  
pre$samples <- rethinking::link(m2\_pre, data = sim\_dat, n = N)  
pre$means <- colMeans(pre$samples)  
pre$PIs <- apply(pre$samples, 2, rethinking::PI)  
# Force the garbage collector to run  
gc()

used (Mb) gc trigger (Mb) max used (Mb)  
Ncells 1897009 101.4 3528810 188.5 3528810 188.5  
Vcells 249518055 1903.7 777212660 5929.7 700336413 5343.2

post <- list()  
post$samples <- rethinking::link(m2\_post, data = sim\_dat, n = N)  
post$means <- colMeans(post$samples)  
post$PIs <- apply(post$samples, 2, rethinking::PI)  
# Add the simulations to the dataset -- this code looks silly but I couldn't  
# think of a better way  
preds\_m2 <-  
 dplyr::bind\_cols(  
 dplyr::bind\_rows(  
 sim\_dat,  
 sim\_dat  
 ),  
 dplyr::bind\_rows(  
 "1" = clean\_list(pre),  
 "2" = clean\_list(post),  
 .id = "t"  
 )  
 )

preds\_m2 |>  
 dplyr::mutate(  
 t = factor(t, levels = c("1", "2"), labels = c("pre", "post"))  
 ) |>  
 ggplot() +  
 aes(  
 x = x, y = est,  
 group = paste(id, t),  
 color = t  
 ) +  
 geom\_line(alpha = 0.05) +  
 labs(  
 x = "Simulated antigenic distance",  
 y = "Counterfactual titer",  
 color = NULL,  
 title = "2017 season: H1N1-Michigan-2015 vaccine",  
 subtitle = "Adaptive pooling; no correlated parameters"  
 ) +  
 scale\_color\_manual(values = c("#E69F00", "#56B4E9")) +  
 guides(color = guide\_legend(override.aes = list(alpha = 1))) +  
 zlib::theme\_ms() +  
 theme(  
 plot.title = element\_text(hjust = 0, margin = margin(0, 0, 5, 0))  
 )



Marginal CIs for the partial pooling model

# Marginalize pre  
pre\_posterior <- rethinking::extract.samples(m2\_pre)  
a\_bar <- rowMeans(pre\_posterior$a)  
b\_bar <- rowMeans(pre\_posterior$b)  
raw\_pre <- sapply(seq(0, 1, 0.01), function(x) pred\_indiv(x, a\_bar, b\_bar))  
pre\_mu <- colMeans(raw\_pre)  
pre\_PI <- apply(raw\_pre, 2, rethinking::PI)  
# Marginalize post  
post\_posterior <- rethinking::extract.samples(m2\_post)  
a\_bar <- rowMeans(post\_posterior$a)  
b\_bar <- rowMeans(post\_posterior$b)  
raw\_post <- sapply(seq(0, 1, 0.01), function(x) pred\_indiv(x, a\_bar, b\_bar))  
post\_mu <- colMeans(raw\_post)  
post\_PI <- apply(raw\_post, 2, rethinking::PI)  
# Make the data frame  
means\_dat\_m2 <-  
 tibble::tibble(  
 x = c(seq(0, 1, 0.01), seq(0, 1, 0.01)),  
 est = c(pre\_mu, post\_mu),  
 lwr = c(pre\_PI[1, ], post\_PI[1, ]),  
 upr = c(pre\_PI[2, ], post\_PI[2, ]),  
 time = factor(  
 rep(c("pre", "post"), each = length(pre\_mu)),  
 levels = c("pre", "post")  
 )  
 )

p2 <- preds\_m2 |>  
 dplyr::mutate(  
 t = factor(t, levels = c("1", "2"), labels = c("pre", "post"))  
 ) |>  
 ggplot() +  
 aes(  
 x = x, y = est,  
 group = paste(id, t),  
 color = t  
 ) +  
 geom\_line(alpha = 0.05) +  
 geom\_ribbon(  
 data = means\_dat\_m2,  
 aes(x = x, ymin = lwr, ymax = upr, y = est, fill = time),  
 alpha = 0.5,  
 inherit.aes = FALSE,  
 show.legend = FALSE  
 ) +  
 geom\_line(  
 data = means\_dat\_m2,  
 aes(x = x, y = est, color = time),  
 linewidth = 0.95, alpha = 1,  
 inherit.aes = FALSE  
 ) +  
 labs(  
 x = "\nSimulated antigenic distance",  
 y = "\nCounterfactual titer\n",  
 color = NULL,  
 subtitle = "Adaptive (partial) pooling"  
 ) +  
 scale\_color\_manual(values = rev(c("#E69F00", "#56B4E9"))) +  
 scale\_fill\_manual(values = rev(c("#E69F00", "#56B4E9"))) +  
 scale\_y\_continuous(  
 limits = c(-2.5, 9),  
 breaks = seq(-2, 8, 2)  
 ) +  
 guides(color = guide\_legend(override.aes = list(alpha = 1))) +  
 zlib::theme\_ms() +  
 coord\_cartesian(expand = FALSE) +  
 theme(  
 plot.title = element\_text(hjust = 0, margin = margin(0, 0, 5, 0))  
 )

## Complete pooling

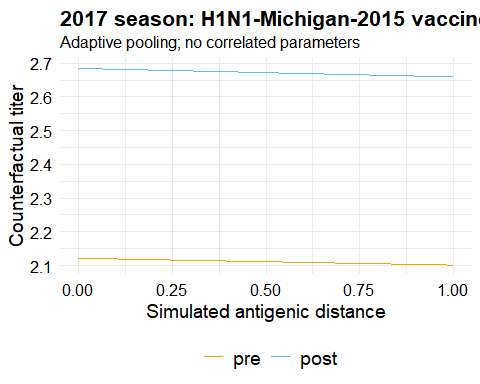
Repeat everything with complete pooling model

set.seed(12312)  
N <- 1000  
m3\_pre <-  
 rethinking::ulam(  
 alist(  
 y ~ dnorm(mu, s),  
 mu <- a + b \* x,  
 a ~ dnorm(5, 5),  
 b ~ dnorm(0, 5),  
 s ~ dexp(1)  
 ),  
 data = d\_pre,  
 chains = 4,  
 cores = 4,  
 iter = N  
 )  
m3\_post <-  
 rethinking::ulam(  
 alist(  
 y ~ dnorm(mu, s),  
 mu <- a + b \* x,  
 a ~ dnorm(5, 5),  
 b ~ dnorm(0, 5),  
 s ~ dexp(1)  
 ),  
 data = d\_post,  
 chains = 4,  
 cores = 4,  
 iter = N  
 )

Get the simulations

# Force the garbage collector to run  
gc()  
pre <- list()  
pre$samples <- rethinking::link(m3\_pre, data = sim\_dat, n = N)  
pre$means <- colMeans(pre$samples)  
pre$PIs <- apply(pre$samples, 2, rethinking::PI)  
# Force the garbage collector to run  
gc()  
post <- list()  
post$samples <- rethinking::link(m3\_post, data = sim\_dat, n = N)  
post$means <- colMeans(post$samples)  
post$PIs <- apply(post$samples, 2, rethinking::PI)  
# Add the simulations to the dataset -- this code looks silly but I couldn't  
# think of a better way  
preds\_m3 <-  
 dplyr::bind\_cols(  
 dplyr::bind\_rows(  
 sim\_dat,  
 sim\_dat  
 ),  
 dplyr::bind\_rows(  
 "1" = clean\_list(pre),  
 "2" = clean\_list(post),  
 .id = "t"  
 )  
 )

preds\_m3 |>  
 dplyr::mutate(  
 t = factor(t, levels = c("1", "2"), labels = c("pre", "post"))  
 ) |>  
 ggplot() +  
 aes(  
 x = x, y = est,  
 group = paste(id, t),  
 color = t  
 ) +  
 geom\_line(alpha = 0.05) +  
 labs(  
 x = "Simulated antigenic distance",  
 y = "Counterfactual titer",  
 color = NULL,  
 title = "2017 season: H1N1-Michigan-2015 vaccine",  
 subtitle = "Adaptive pooling; no correlated parameters"  
 ) +  
 scale\_color\_manual(values = c("#E69F00", "#56B4E9")) +  
 guides(color = guide\_legend(override.aes = list(alpha = 1))) +  
 zlib::theme\_ms() +  
 theme(  
 plot.title = element\_text(hjust = 0, margin = margin(0, 0, 5, 0))  
 )



Marginal CIs for the partial pooling model

# Marginalize pre  
pre\_posterior <- rethinking::extract.samples(m3\_pre)  
pre\_posterior <- lapply(pre\_posterior, \(x) matrix(x, ncol = 1))  
a\_bar <- rowMeans(pre\_posterior$a)  
b\_bar <- rowMeans(pre\_posterior$b)  
raw\_pre <- sapply(seq(0, 1, 0.01), function(x) pred\_indiv(x, a\_bar, b\_bar))  
pre\_mu <- colMeans(raw\_pre)  
pre\_PI <- apply(raw\_pre, 2, rethinking::PI)  
# Marginalize post  
post\_posterior <- rethinking::extract.samples(m3\_post)  
post\_posterior <- lapply(post\_posterior, \(x) matrix(x, ncol = 1))  
a\_bar <- rowMeans(post\_posterior$a)  
b\_bar <- rowMeans(post\_posterior$b)  
raw\_post <- sapply(seq(0, 1, 0.01), function(x) pred\_indiv(x, a\_bar, b\_bar))  
post\_mu <- colMeans(raw\_post)  
post\_PI <- apply(raw\_post, 2, rethinking::PI)  
# Make the data frame  
means\_dat\_m3 <-  
 tibble::tibble(  
 x = c(seq(0, 1, 0.01), seq(0, 1, 0.01)),  
 est = c(pre\_mu, post\_mu),  
 lwr = c(pre\_PI[1, ], post\_PI[1, ]),  
 upr = c(pre\_PI[2, ], post\_PI[2, ]),  
 time = factor(  
 rep(c("pre", "post"), each = length(pre\_mu)),  
 levels = c("pre", "post")  
 )  
 )

p3 <- preds\_m3 |>  
 dplyr::mutate(  
 t = factor(t, levels = c("1", "2"), labels = c("pre", "post"))  
 ) |>  
 ggplot() +  
 aes(  
 x = x, y = est,  
 group = paste(id, t),  
 color = t  
 ) +  
 geom\_line(alpha = 0.05) +  
 geom\_ribbon(  
 data = means\_dat\_m3,  
 aes(x = x, ymin = lwr, ymax = upr, y = est, fill = time),  
 alpha = 0.5,  
 inherit.aes = FALSE,  
 show.legend = FALSE  
 ) +  
 geom\_line(  
 data = means\_dat\_m3,  
 aes(x = x, y = est, color = time),  
 linewidth = 0.95, alpha = 1,  
 inherit.aes = FALSE  
 ) +  
 labs(  
 x = "\nSimulated antigenic distance",  
 y = "\nCounterfactual titer\n",  
 color = NULL,  
 subtitle = "Complete pooling"  
 ) +  
 scale\_color\_manual(values = rev(c("#E69F00", "#56B4E9"))) +  
 scale\_fill\_manual(values = rev(c("#E69F00", "#56B4E9"))) +  
 scale\_y\_continuous(  
 limits = c(-2.5, 9),  
 breaks = seq(-2, 8, 2)  
 ) +  
 guides(color = guide\_legend(override.aes = list(alpha = 1))) +  
 zlib::theme\_ms() +  
 coord\_cartesian(expand = FALSE) +  
 theme(  
 plot.title = element\_text(hjust = 0, margin = margin(0, 0, 5, 0))  
 )

I think somewhere I broke the CIs for the code in this document, but they will be fixed in the next one. The function to calculate them works, but this code is old and doesn’t use the rethinking helpers functions I wrote.

# Comparison

|  |
| --- |
|  |

These are the “type 1” credible intervals from Andrew Heiss’ blog post (allegedly the CIs for the global mean). The type 3 credible intervals would be much larger. These also use link instead of sim so they are credible intervals for the conditional mean, not for the individual outcome.