bayesParcel

Final project for Bayesian Data Analaysis course

Yaniv Abir, Paul Bloom, Monica Thieu

Fall 2018

Contents

Introduction

Functional magnetic resonance imaging (fMRI) is among the most popular methods used by psychologists and cognitive neuroscience to examine human brain activity. Because brain images for fMRI analysis contain hundreds of thousands of voxels (3-dimensional pixels), multiple comparisons are a large issue when attempting to make inference using this type of data. Over the last several years, the MRI field in general has become more aware of these issues, as the power and replicability of many MRI studies has been shown often to be quite low (Geuter et al., 2018; Button et al., 2013; Lieberman & Cunningham, 2009).

One reason in particular why MRI analyses can tend to be low-powered is that voxel-level data is extremely noisy and unreliable. When conducting analyses to make inferences about populations, researchers typically first estimate subject-level activity at each voxel (we do not focus on these subject-level estimates for the purposes of this analysis), then when making inference on groups of subjects, use strict corrections in an attempt to control Type 1 errors in identifying which individual voxels are "active" under a null hypothesis significance testing (NHST) framework (Genovese, Lazar, & Nichols, 2002). This method of running a corrected test at the population level on each voxel independently, known as the 'massively univariate' approach to fMRI analysis, is currently an extremely popular tool.

Alternatively, if one has a prior hypothesis about a specific brain region for a population-level effect, it is common to use the 'region of interest' (ROI) approach, where population inference is done after voxelwise estimates are averaged across a spatial region of voxels. While this approach assumes that voxels within an ROI serve as one functional unit, because only one region is being investigated, power to detect true effects is (at least theoretically) higher. In practice, however, researchers may have many degrees of freedom in

selecting ROIs, and estimates are biased if labs make it a practice of searching many regions (effectively making many comparisons) using NHST.

Inspired by ideas from a recent preprint (Chen et al., 2018) we set out here to test an alternate approach to whole-brain inference by employing Bayesian Hierarchical Modeling with ROIs as the observational units. This approach should let us estimate activity for all regions at the population level simultaneously, using partial pooling to effectively account for multiple comparisons (Gelman, Hill, & Yajima, 2012). Under this approach, we model the predicted activity for each region, within each subject. Similar to Gelman et al.'s (2013) "8 schools" example, we have data in the form of an estimate and standard deviation for each ROI, for each subject. ROIs in this case are defined using 68 regions of structural/functional signifance from the Harvard-Oxford Brain Atlas,

Crucial to our models are the hyperparameters for the distributions from which different regions and subjects are drawn from. To address this, we try two models:

- Model 0: This model assumes that all ROIs, at the population level, are drawn from one common distribution with a prior centered around 0 (or no activity).
- Model 1: This model attempts to classify ROIs into mixture components of 'positively active', 'inactive', and 'negatively active' rather than assuming ROIs area all drawn from a common distribution.

Below, we demonstrate both models, and address their efficacy on both simulated and real data.

Dataset for this case study

For this example, we test our model on an emotional face categorization task scanned using fMRI on a group of healthy developing children. In the task, participants are shown different emotional faces and asked to press buttons to categorize which emotion they are seeing (neutral versus fearful faces). For our purposes, we focuse on estimating the activity in the brain associated with viewing the fearful faces above the 'baseline' of a fixation cross on the screen. For more information on this task, see (Gee et al., 2013)

Similarly to the 8 schools example, we have a single estimate for fearful faces for each ROI for each subject. In addition, we have a standard deviation for each of these estimates. Our data are formatted as below:

```
betas_raw <- read_csv("../../ignore/harvard_ox_roi.csv")

# nonsense ROIs for excluding from analysis
exclude <- c('harvardox_subcortical_1', 'harvardox_subcortical_2', 'harvardox_subcortical_8',
             'harvardox_subcortical_12', 'harvardox_subcortical_13')

betas <- betas_raw %>%
  gather(key = "roi", value = "value", -Subject, -wave, -meanFD_included_trs) %>%
  filter(!is.na(value)) %>%
  mutate(statistic = if_else(grepl("Mean", roi), "cope_mean", "cope_sd"),
         roi = if_else(statistic == "cope_mean",
                       stringr::str_sub(roi, end = -5L),
                       stringr::str_sub(roi, end = -3L))) %>%
  spread(statistic, value) %>%
  # Filter out cope sd's that are 0 -- this would indicate an error in preprocessing
  filter(cope_sd > 0,
         wave == 1,
         # Exclude nonsense ROIs
         !(roi %in% exclude)) %>%
  group_by(Subject) %>%
  nest() %>%
  mutate(subject_num = 1:n()) %>%
  unnest() %>%
```

```
  group_by(roi) %>%
  nest() %>%
  mutate(roi_num = 1:n()) %>%
  unnest() %>%
  mutate(cope_mean_scaled = cope_mean / sd(cope_mean),
         cope_sd_scaled = cope_sd / sd(cope_mean))

head(betas)
```

```
## # A tibble: 6 x 10
##   roi   roi_num Subject subject_num  wave meanFD_included~ cope_mean
##   <chr>   <int>   <int>       <int> <int>            <dbl>     <dbl>
## 1 harv~       1       1           1     1           0.112     401.
## 2 harv~       1       2           2     1           0.126    -123.
## 3 harv~       1       3           3     1           0.118     253.
## 4 harv~       1       7           4     1           0.165     521.
## 5 harv~       1       8           5     1           0.0945      7.07
## 6 harv~       1      18           6     1           0.218      26.0
## # ... with 3 more variables: cope_sd <dbl>, cope_mean_scaled <dbl>,
## #   cope_sd_scaled <dbl>
```

Model 0: Single normal hyper-distribution

As a starting point, we chose to model COPE responses in the different ROIs with a model based on the 8-school example. As such, this is a hierarchical model, with each ROI drawn from a population of ROIs with a normal distribution, and each subject drawn from a population of subjects with a normal distribution. Thus, the likelihood in our model is:

$$COPE_i \sim normal(\alpha + \tau_{ROI} * \eta_{ROI[i]} + \tau_{subject} * \eta_{subject[i]}, \sigma_{COPE})$$

With α being a general intercept term, ηs as the ROI- and subject-wise deviations from the general intercept, and τs as the variation factor for the deviations. As in the 8 school model, we assume that σ_{COPE} are known. Following the 8-school model, we chose normal mid-level priors for ROIs and subjects:

$$\eta_{subject} \sim normal(0, 1)$$

$$\eta_{ROI} \sim normal(0, 1)$$

Finally, we used non-informative priors for the variance and general expected value hyper parameters (α and $\tau_{subject/ROI}$), for regularization purposes.

Stan code

Fake data simulation

Before fitting the model to the data, we'll begin by evaluating the model's potency in recovering known parameters. We used Stan to simulate data given parameters drawn from the priors in our model. We then fit the simulated data, and checked the resultant posterior intervals against the known values of the parameters.

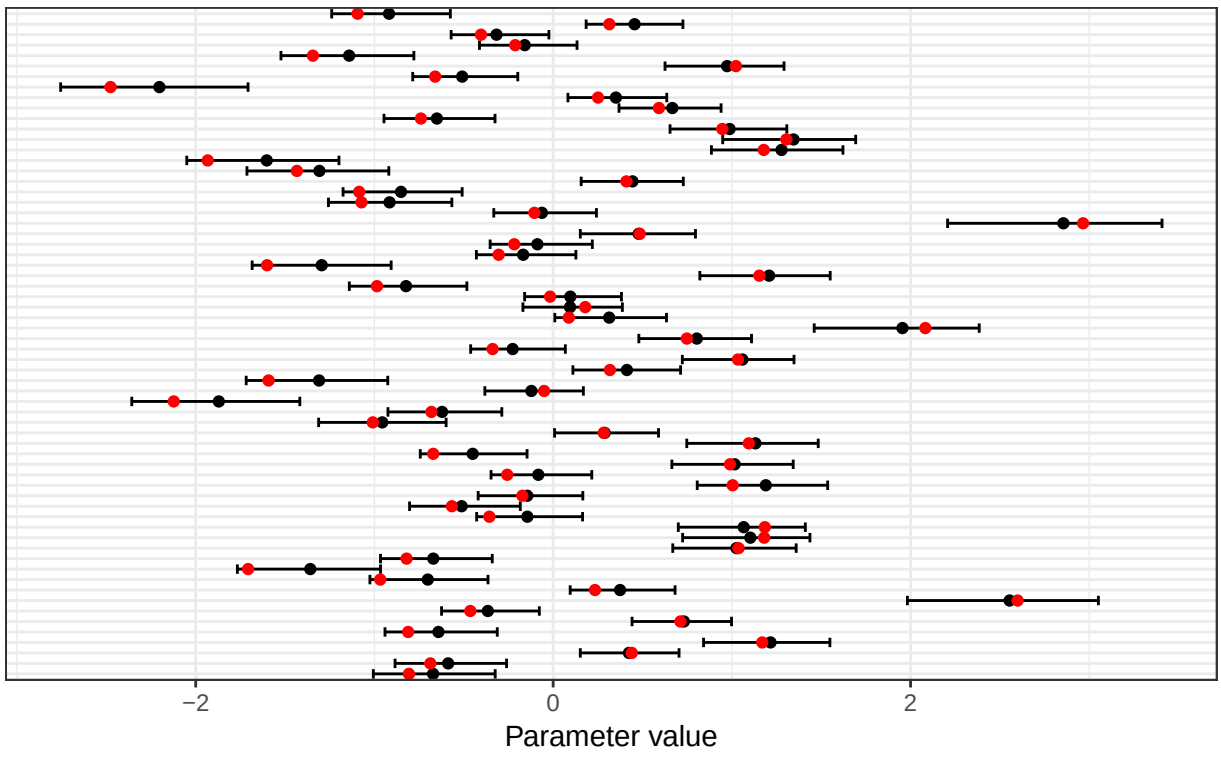
```
# Draw fake data
model_fake_data0 <- stan("../stan/model0_fake.stan",
                    data = betas %$%
                      list(N = nrow(.),
```

```
                          N_roi = max(roi_num),
                          N_subj = max(subject_num),
                          roi = roi_num,
                          subj = subject_num,
                          fd = meanFD_included_trs,
                          varcope = cope_sd_scaled),
                   iter = 1, warmup = 0, chain = 1, seed = 78)

# Fake data book keeping
fake_params0 <- model_fake_data0 %>%
  as.data.frame() %>%
  as_tibble() %>%
  select(-contains("cope"))

fake_data0 <- model_fake_data0 %>%
  as.data.frame() %>%
  as_tibble() %>%
  select(contains("cope")) %>%
  gather() %>%
  # implicit ordering of the simulated data should be in the same order as real data
  bind_cols(betas %>%
              select(roi_num, subject_num, meanFD_included_trs, cope_sd_scaled)) %>%
  select(-key)

save(fake_data0, fake_params0, file = "../../ignore/fake_data0.rda")
```

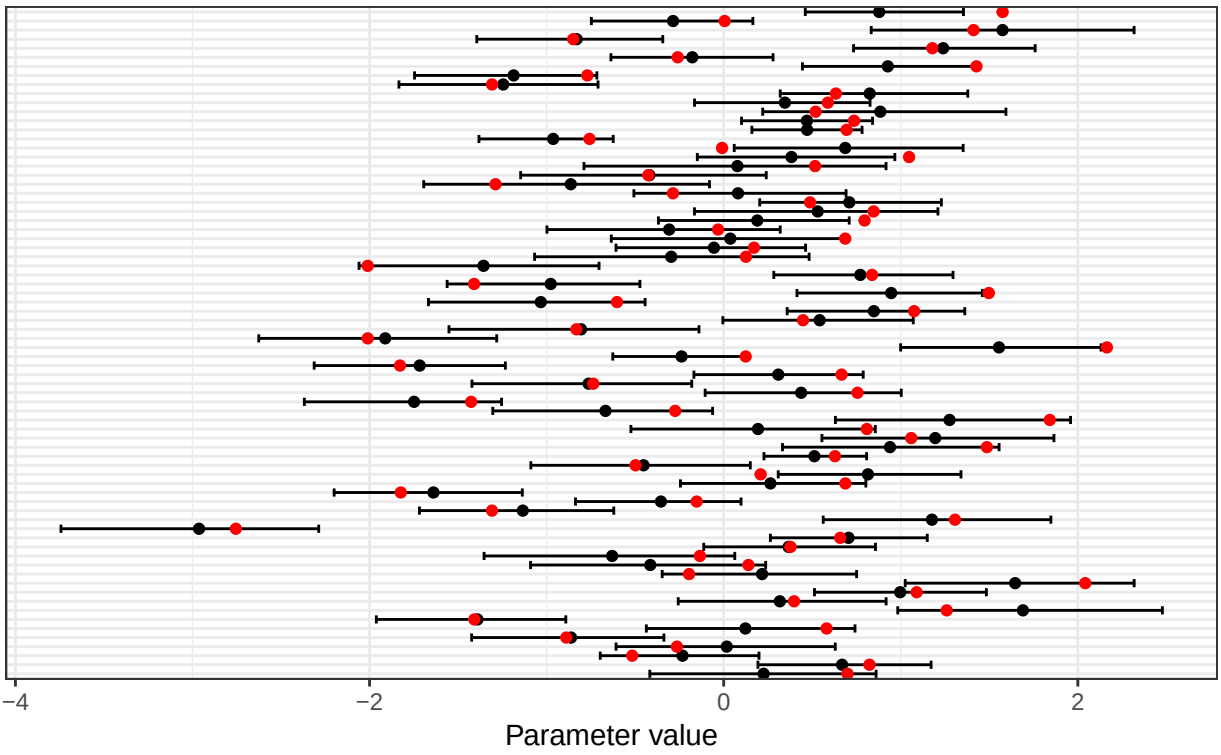
```
# Fit fake data
fake_fit0 <- stan("../stan/model0.stan",
                  data = betas %$%
                    list(N = nrow(.),
                         N_roi = max(roi_num),
                         N_subj = max(subject_num),
                         cope = fake_data0$value,
                         roi = roi_num,
                         subj = subject_num,
                         fd = meanFD_included_trs,
                         varcope = cope_sd_scaled),
                  seed = 2323)
save(fake_fit0, file = "../../ignore/fake_fit0.rda")
```

```
# Check parameter recovery
fake_fit0 <- as.data.table(fake_fit0)
fake_fit0 <- melt(fake_fit0, measure.vars = colnames(fake_fit0))
fake_fit0 <- fake_fit0[, .(median = median(value),
                           ub = quantile(value, 0.975),
                           lb = quantile(value, 0.025)), by = variable]
fake_params0 <- melt(fake_params0, measure.vars = colnames(fake_params0), value.name = 'trueVal')
fake_fit0 <- merge(fake_fit0, fake_params0, by = 'variable')
fake_fit0 <- fake_fit0[!grepl('lp_', variable, fixed = T),]

ggplot(fake_fit0[grepl('eta_roi',variable),], aes(x = median, y = variable)) +
  geom_errorbarh(aes(xmin = lb, xmax = ub)) +
  geom_point() +
  geom_point(aes(x = trueVal), color = 'red') +
```

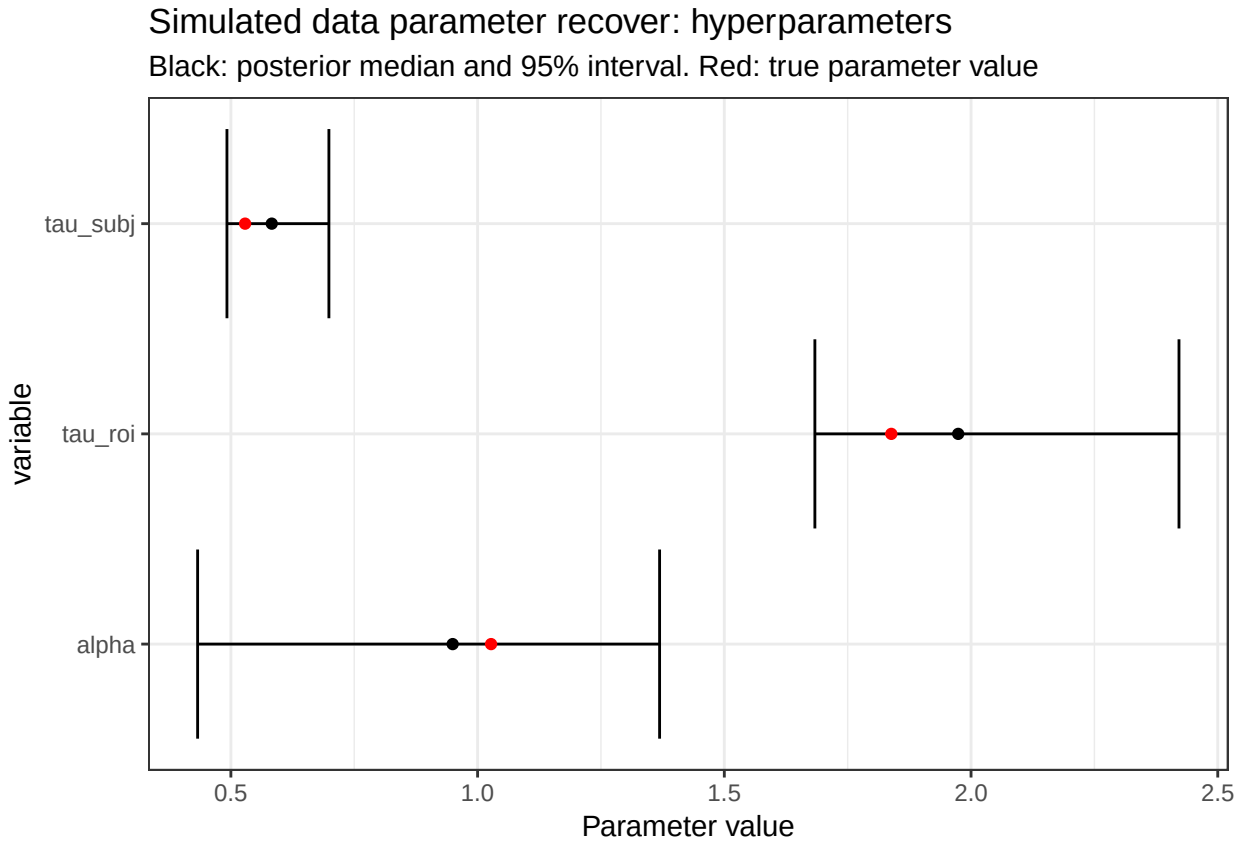
```
  theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank()) +
  labs(x = 'Parameter value',
       title = 'Simulated data parameter recover: eta_ROI',
       subtitle = 'Black: posterior median and 95% interval. Red: true parameter value')
```




```
ggplot(fake_fit0[grepl('eta_subj',variable),], aes(x = median, y = variable)) +
  geom_errorbarh(aes(xmin = lb, xmax = ub)) +
  geom_point() +
  geom_point(aes(x = trueVal), color = 'red') +
  theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank()) +
  labs(x = 'Parameter value',
       title = 'Simulated data parameter recover: eta_subject',
       subtitle = 'Black: posterior median and 95% interval. Red: true parameter value')
```




```
ggplot(fake_fit0[!grepl('eta_roi',variable) &!grepl('eta_subj',variable),],
       aes(x = median, y = variable)) +
  geom_errorbarh(aes(xmin = lb, xmax = ub)) +
  geom_point() +
  geom_point(aes(x = trueVal), color = 'red') +
  labs(x = 'Parameter value',
       title = 'Simulated data parameter recover: hyperparameters',
       subtitle = 'Black: posterior median and 95% interval. Red: true parameter value')
```



As can be seen in the plots above, for the vast majority of parameters in this fit, the true value lies within the 95% posterior interval, as we would expect. Since we drew parameters from the priors for our model, we can see this result as rather general.

Model fit to data

Next, we fit the observed data with our model.

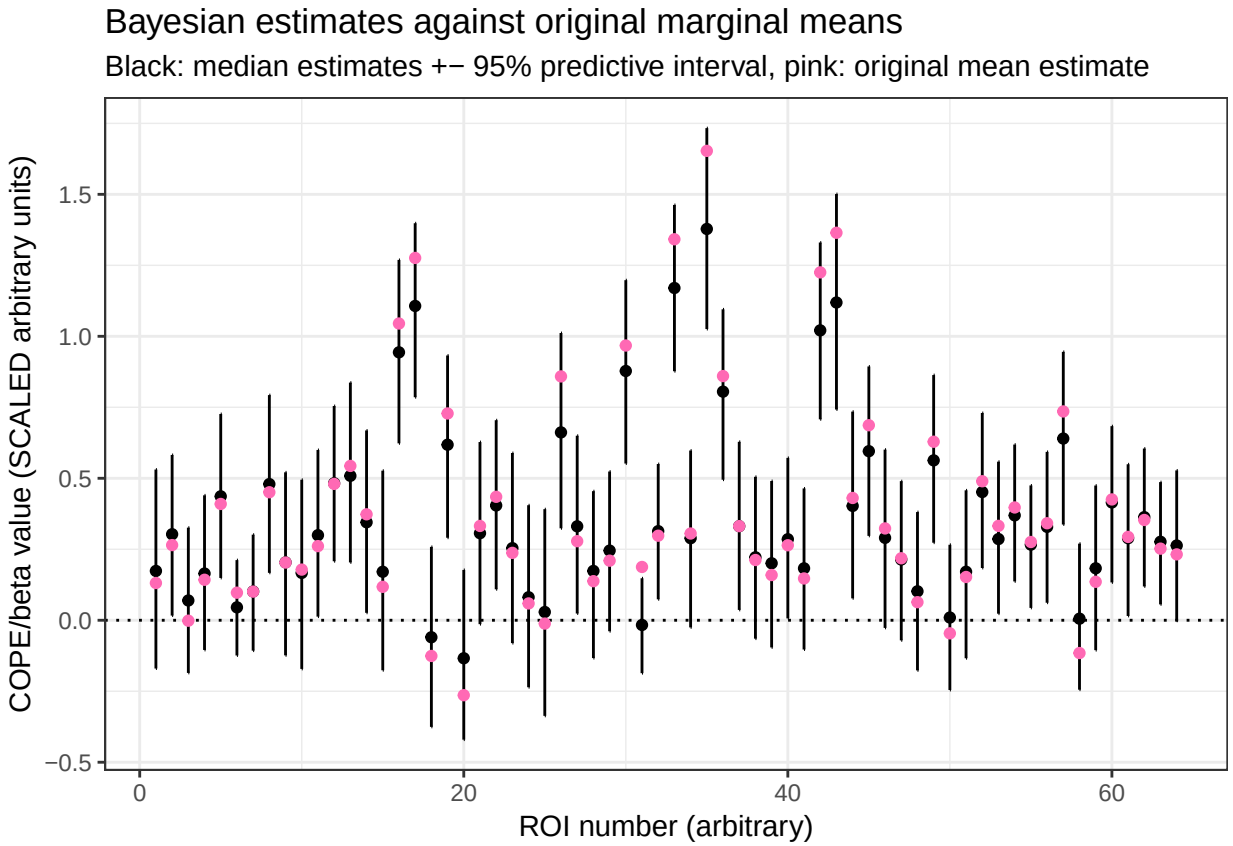
```
fit0 <- stan("../stan/model0.stan",
                  data = betas %$%
                    list(N = nrow(.),
                         N_roi = max(roi_num),
                         N_subj = max(subject_num),
                         cope = cope_mean_scaled,
                         roi = roi_num,
                         subj = subject_num,
                         fd = meanFD_included_trs,
                         varcope = cope_sd_scaled),
                  control = list(max_treedepth = 15),
                  seed = 909)

save(fit0, file = "../../ignore/fit_model0.rda")
```

```
df_fit0 <- fit0 %>%
  as.data.frame() %>%
  as_tibble() %>%
  mutate(iteration = 1:n())
```

```
hyperparams <- df_fit0 %>%
  select(iteration, alpha, tau_roi, tau_subj)

theta_rois <- df_fit0 %>%
  select(iteration, starts_with("theta_roi")) %>%
  gather(key = "roi_num", value = "theta", starts_with("theta_roi")) %>%
  mutate(roi_num = as.integer(str_sub(roi_num, start = 11L, end = -2L)),
         theta_unscaled = theta * sd(betas$cope_mean)) %>%
  group_by(roi_num) %>%
  nest(.key = "iterations") %>%
  left_join(betas %>%
              select(roi_num, cope_mean, cope_mean_scaled) %>%
              group_by(roi_num) %>%
              summarize_all(mean),
            by = "roi_num") %>%
  mutate(summaries = map(iterations, ~.x %>%
                           summarize_at(vars(starts_with("theta")),
                                        funs(median = median,
                                             int_95_lower = quantile(., .025),
                                             int_95_upper = quantile(., .975),
                                             int_50_lower = quantile(., .25),
                                             int_50_upper = quantile(., .75))))) %>%
  unnest(summaries, .preserve = "iterations")
```

After fitting the model, we proceeded to perform some model check and evaluations. Comparing the observed COPE values to the estimated posterior intervals, we find that the estimated intervals are pooled towards the sample mean, in comparison to the raw values. The plot below illustrates this point. This is an expected advantageous feature of the hierarchical model, which allows partial pooling of estimates across ROIs and subjects.

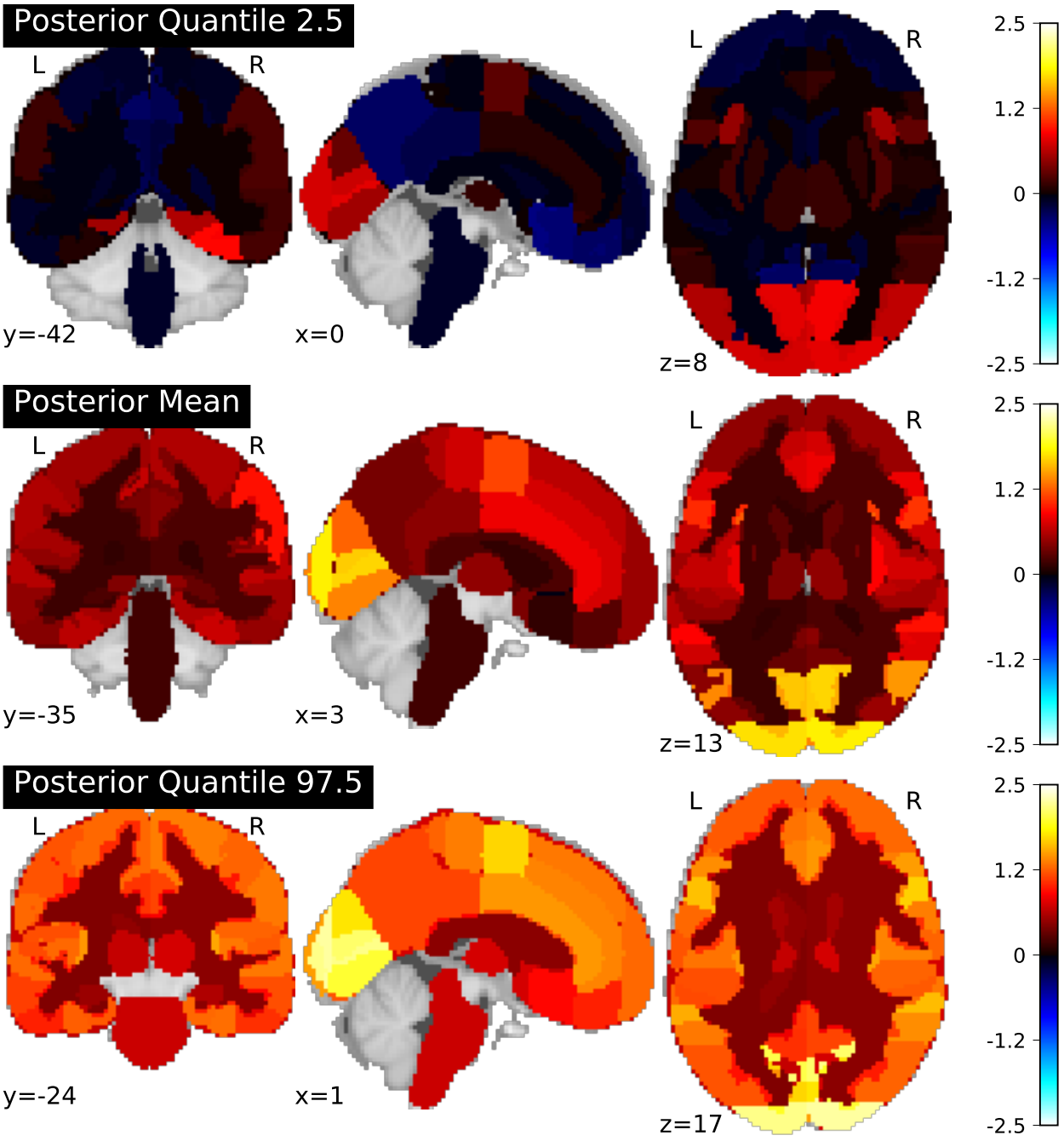
```
theta_rois %>%
  ggplot(aes(x = roi_num)) +
  geom_hline(yintercept = 0, linetype = 3) +
  geom_errorbar(aes(ymin = theta_int_95_lower, ymax = theta_int_95_upper), width = 0) +
  geom_point(aes(y = theta_median)) +
  geom_point(aes(y = cope_mean_scaled), color = "hotpink") +
  labs(x = "ROI number (arbitrary)",
       y = "COPE/beta value (SCALED arbitrary units)",
       title = "Bayesian estimates against original marginal means",
       subtitle = "Black: median estimates +- 95% predictive interval, pink: original mean estimate") +
  theme_bw()
```



Brain plots

Here, we plot our posterior estimated betas for each ROI at the group level back onto the brain to see whether the results look reasonable. We can check the results of these betas from this model against a massively univariate model just as a sanity check – are our models demonstrating similar patterns as standard fMRI processing software (FSL)?

Let's first look at the posterior mean for each ROI, as well as 2.5% and 97.5% quantiles of the posterior.



This looks sensible! We are seeing the highest estimates mostly in the occipital lobe at primary visual cortex where early visual processing is thought to take place. That makes sense. Interestingly, in this task contrast (fearful faces over baseline), almost all regions of the brain seem to be estimated as being positively, rather than negatively, active. This makes some sense, as we'd expect most brain regions to be positively activated by a task over baseline (we would see more negative patterns if we contrasted one task versus another).

Let's compare it to the univariate map, which plots a T-statistic for activity for every voxel.

This looks fairly similar, so that's good! We see the same pattern of highest activity in the primary visual cortex and similar patterns in general across the rest of the brain. One interesting difference is that we some negative estimates in the mass univariate approach, particularly in the prefrontal cortex, that don't show up in our model. This could be because our model is pooling the most negative estimates more positively –

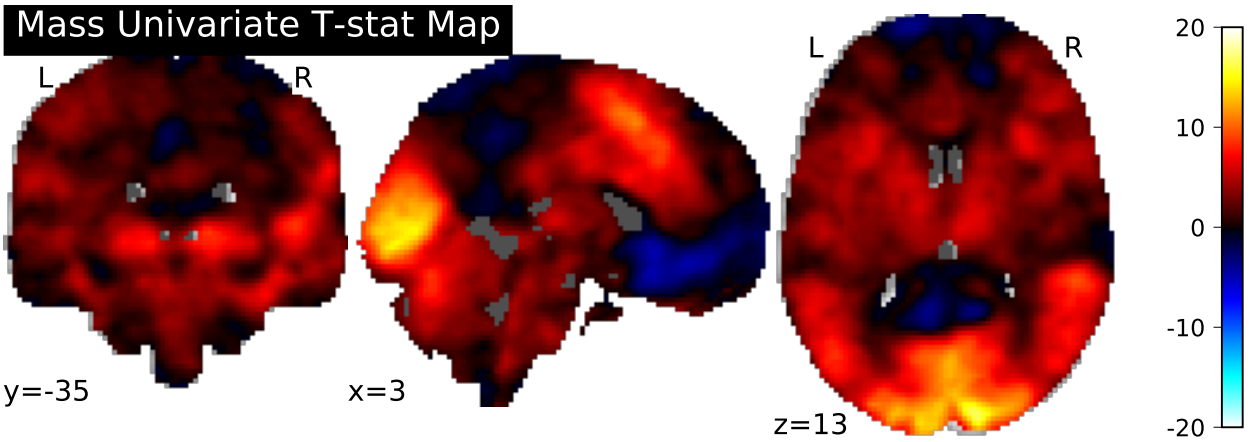


Figure 1:

the areas that show up as negative in the mass univariate map do seem to correspond with more negatively estimated areas in our model. Then, the question would be whether this pooling is appropriate. In one attempt to address this question of whether we were estimating all regions too positively (and therefore needing to allow some to be negative), we decided to try out the mixture model approach to categorize some ROIs as negative.

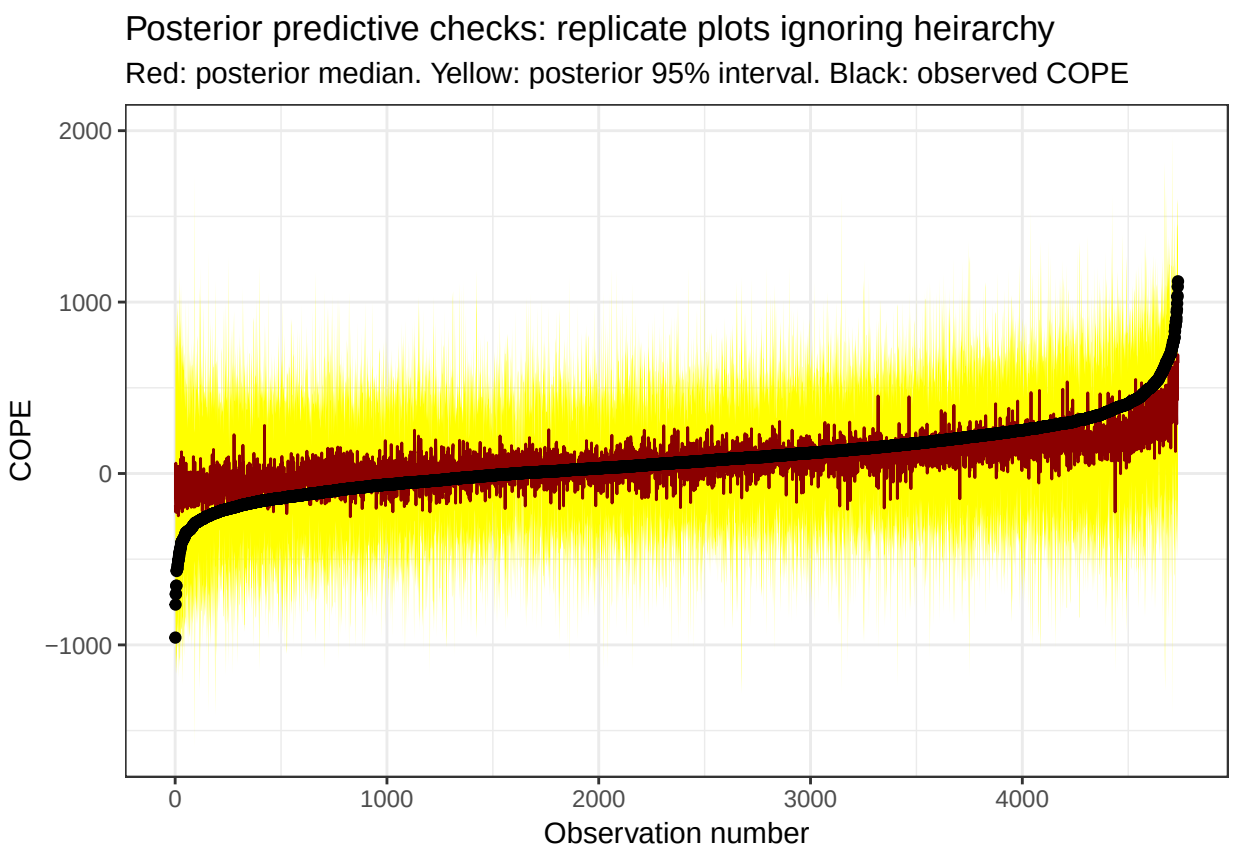
Posterior predictive checks

To further evaluate the adequacy of the hierarchical normal model to the data, we performed posterior predictive checks. For each posterior draw, simulated data were generated with the same model likelihood, given the drawn parameters. These posterior replicates allow us to plot the interval of expected new observations, given our model and the observed data. Comparing our observed data to the predictive interval informs us about the adequacy of the chosen model.

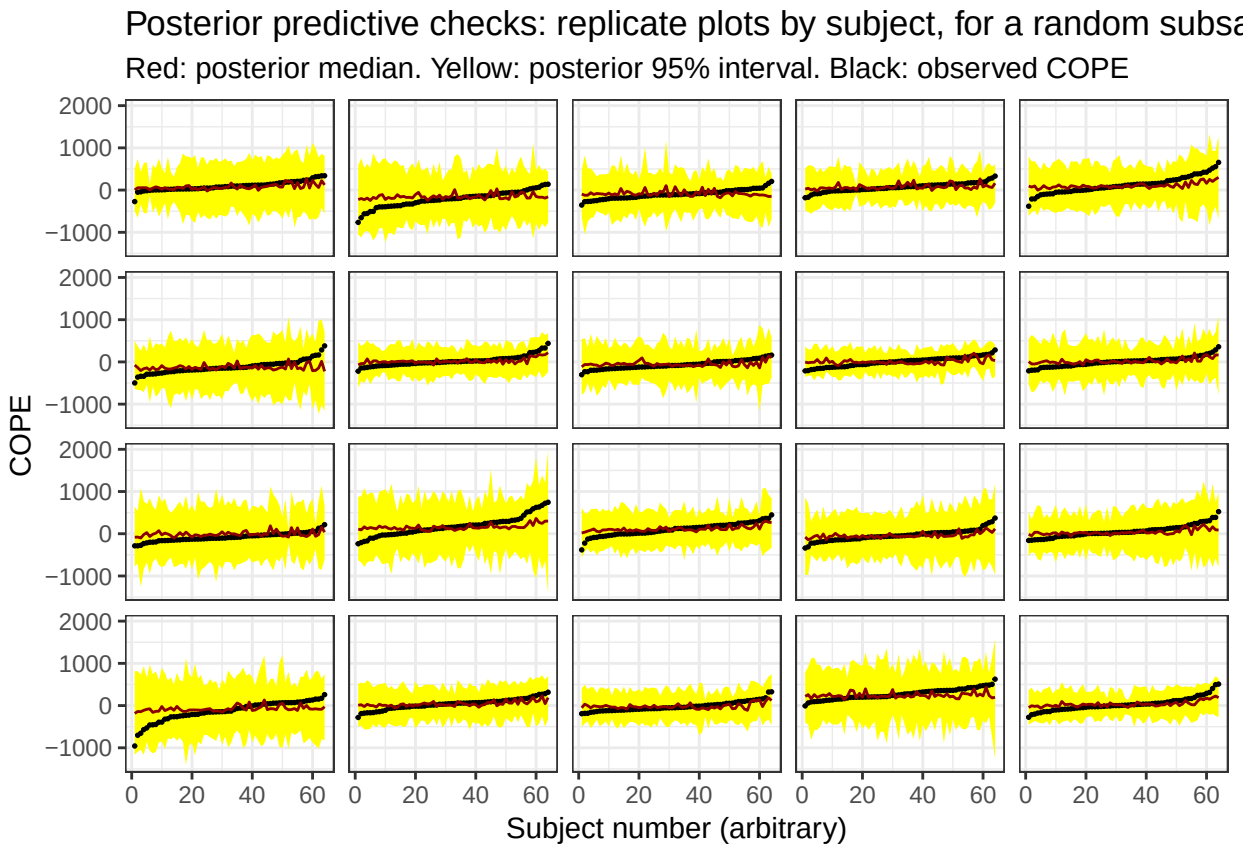
As can be seen of in the plots of replicates and observed data below, the observed data all fall withing the 95% interval of replicates. Thus, the data are plausible given the generative model. A drawback highlighted by this test, is that the median replicates are do not follow the observed data well at the extreme tails of the distribution: that is, the observed data seems to have consistently heavier tails than than the replicate data.

```
# Plot data replicates
# Implemented with data.table. Sorry. Open to the possibility of recoding with tidyverse, just not righ

df_fit0 <- as.data.table(fit0)
df_fit0 <- df_fit0[, grepl('rep_cope', colnames(df_fit0)), with = F]
df_fit0[, sim := 1:nrow(df_fit0)]
df_fit0 <- melt(df_fit0, id.vars = 'sim')
df_fit0[, value := value * sd(betas$cope_mean)]
df_fit0 <- df_fit0[, .(median = median(value),
                       lb = quantile(value, 0.025),
                       ub = quantile(value, 0.975)), by = variable][order(variable)]
df_fit0[, subject_num := betas$subject_num]
df_fit0[, roi_num := betas$roi_num]
df_fit0[, true_value := betas$cope_mean]
df_fit0 <- df_fit0[order(true_value)]

# Plot disregarding hierarchy
ggplot(df_fit0, aes(x = 1:nrow(df_fit0), y = median)) +
```

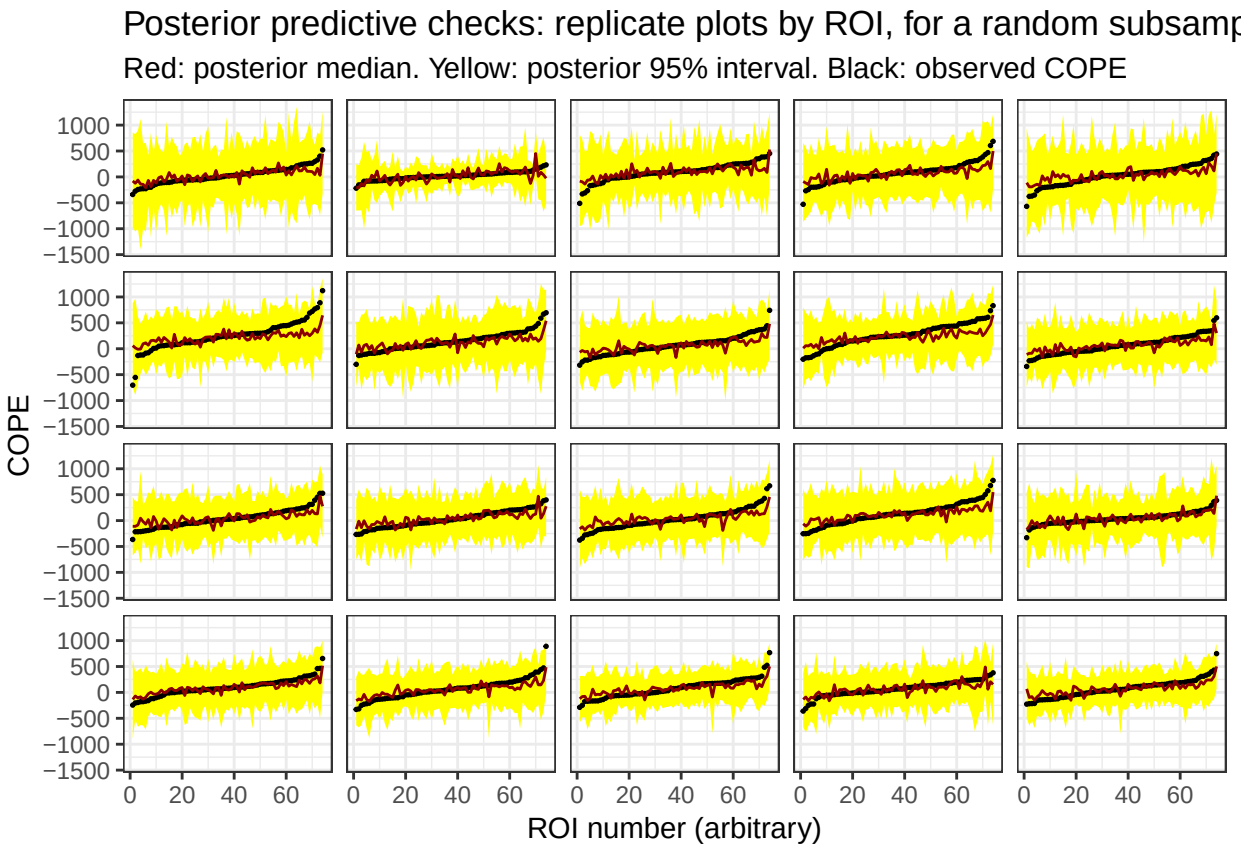
```
  geom_ribbon(aes(ymin = lb, ymax = ub), fill = 'yellow') +
  geom_line(color = 'darkred') +
  geom_point(aes(y = true_value)) +
  labs(x = 'Observation number',
       y = 'COPE',
       title = 'Posterior predictive checks: replicate plots ignoring heirarchy',
       subtitle = 'Red: posterior median. Yellow: posterior 95% interval. Black: observed COPE')
```




```
# Plot by subject
df_subs <- df_fit0[subject_num %in% sample(1:length(unique(subject_num)), 20)]
df_subs[, xplot := 1:.N, by = subject_num]
ggplot(df_subs, aes(x = xplot, y = median)) +
  geom_ribbon(aes(ymin = lb, ymax = ub), fill = 'yellow') +
  geom_point(aes(y = true_value), size = 0.3) +
  geom_line(color = 'darkred') +
  facet_wrap('subject_num') +
  theme(strip.background = element_blank(),
    strip.text.x = element_blank()) +
  labs(x = 'Subject number (arbitrary)',
       y = 'COPE',
       title = 'Posterior predictive checks: replicate plots by subject, for a random subsample of subj
       subtitle = 'Red: posterior median. Yellow: posterior 95% interval. Black: observed COPE')
```



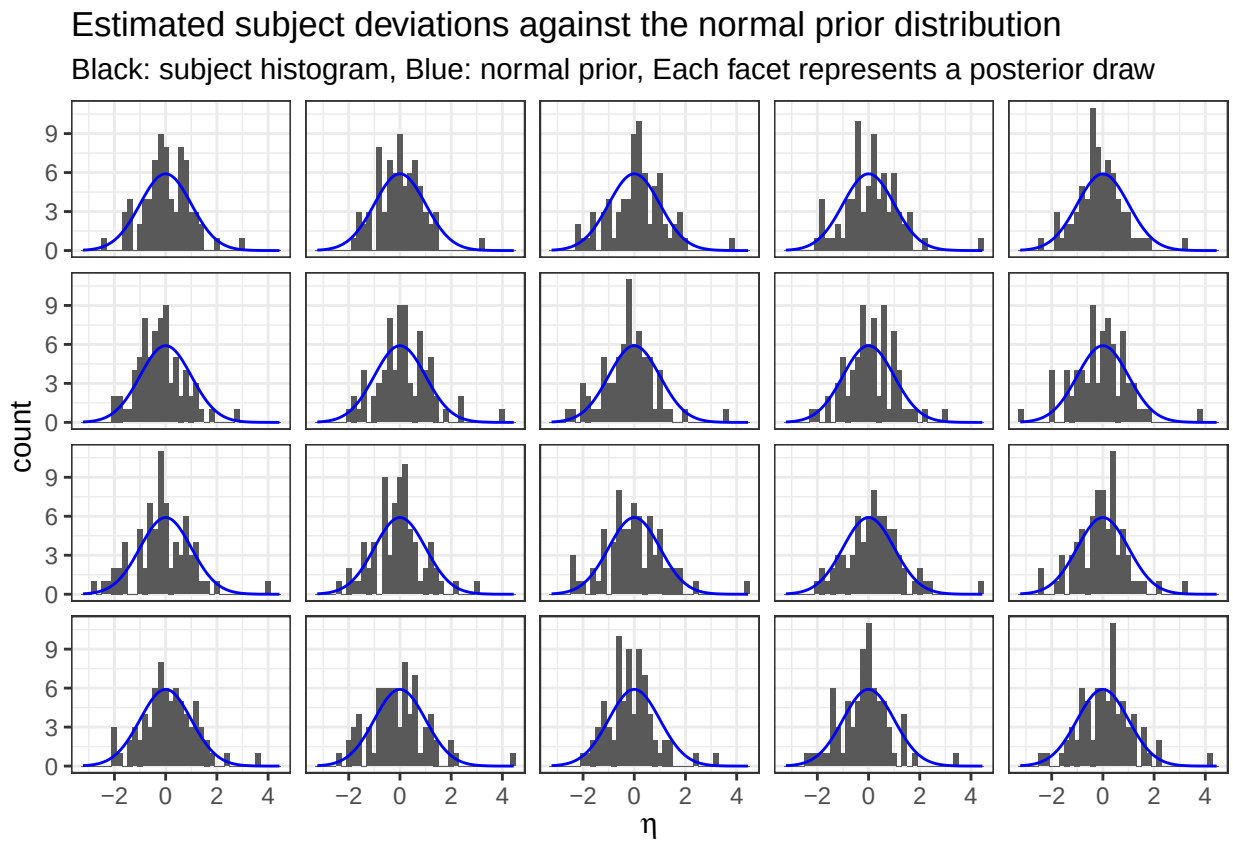

```
# Plot by roi
df_subs <- df_fit0[roi_num %in% sample(1:length(unique(roi_num)), 20)]
df_subs[, xplot := 1:.N, by = roi_num]
ggplot(df_subs, aes(x = xplot, y = median)) +
  geom_ribbon(aes(ymin = lb, ymax = ub), fill = 'yellow') +
  geom_point(aes(y = true_value), size = 0.3) +
  geom_line(color = 'darkred') +
  facet_wrap('roi_num') +
  theme(strip.background = element_blank(),
    strip.text.x = element_blank()) +
  labs(x = 'ROI number (arbitrary)',
       y = 'COPE',
       title = 'Posterior predictive checks: replicate plots by ROI, for a random subsample of ROIs',
       subtitle = 'Red: posterior median. Yellow: posterior 95% interval. Black: observed COPE')
```



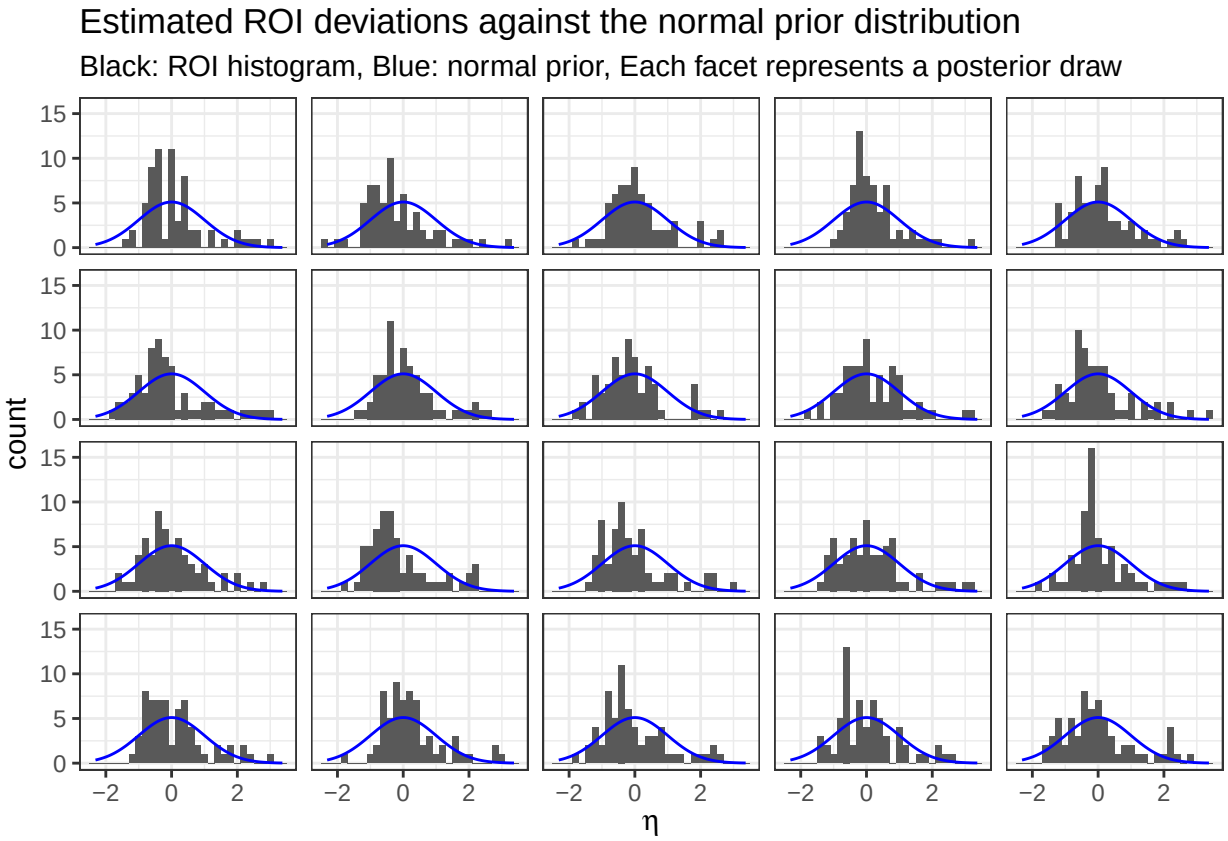
The heavier tails in the observed data indicated that our mid-level priors might not be the best fit for the data. Hence, we compared the estimated draws for the ROI and subject deviations from the mean COPE to the normal prior we assigned them. The plots below reveal that overall, the estimated deviations conform to the shape of the normal prior. However, when inspecting the plots for the ROIs, they seem to have a rightward skew and a heavier right tail that is not consistent with a normal prior.

```
# Compare thetas to priors
df_fit0 <- as.data.table(fit0)
vars <- colnames(df_fit0)
eta_rois <- df_fit0[sample(1:nrow(df_fit0), 20),
                    grepl('eta_roi', vars) & !grepl('th', vars), with = F]
eta_subj <- df_fit0[sample(1:nrow(df_fit0), 20),
                    grepl('eta_subj', vars) & !grepl('th', vars), with = F]

eta_rois[, sim := 1:.N]
eta_subj[, sim := 1:.N]

eta_rois <- melt(eta_rois, id.vars = 'sim')
eta_subj <- melt(eta_subj, id.vars = 'sim')

bw <- .2
n_subj <- length(unique(eta_subj$variable))
ggplot(eta_subj, aes(x = value)) +
  geom_histogram(binwidth = bw) +
  facet_wrap('sim') +
  stat_function(fun = function(x, bw, n) dnorm(x) * bw * n,
                args = c(bw = bw, n = n_subj),
```

```
                color = 'blue') +
  theme(strip.background = element_blank(),
    strip.text.x = element_blank()) +
  labs(x = expression(eta),
       title = 'Estimated subject deviations against the normal prior distribution',
       subtitle = 'Black: subject histogram, Blue: normal prior, Each facet represents a posterior draw
```




```
n_roi <- length(unique(eta_rois$variable))
ggplot(eta_rois, aes(x = value)) +
  geom_histogram(binwidth = bw) +
  facet_wrap('sim') +
  stat_function(fun = function(x, bw, n) dnorm(x) * bw * n,
                args = c(bw = bw, n = n_roi),
                color = 'blue') +
  theme(strip.background = element_blank(),
    strip.text.x = element_blank()) +
  labs(x = expression(eta),
       title = 'Estimated ROI deviations against the normal prior distribution',
       subtitle = 'Black: ROI histogram, Blue: normal prior, Each facet represents a posterior draw')
```



Model 0 summary

Fitting a hierarchical normal model to the data, with normally distributed deviations for ROIs and subjects, seems like a good first choice for the data. The estimates are stable and interpretable, with observed data plausible given the generative model. However, a tendency towards rightward skew in the distribution of estimated ROIs, and the comparison of replicates and observed data, prompted us to suspect that the extreme observations could be better modeled. This is in line with the modal theoretical view of imaging experiments: It is expected that regions in the brain should come from separate distributions. Commonly, it is assumed that a group of regions is positively activated by a given task, another group does not show activation, and possibly a third group shows negative values for the contrast of interest.

Model 1: Finite mixture hyper-distribution

We decided that we would implement this theoretically motivated second model as a finite mixture model. In this case, we would assume one mixture component for "non-activated" regions, one mixture component for "negative-activated" regions, and one for "positive-activated" regions. In this way, we would be able to classify regions as activated vs. not based on their estimated probabilities of belonging to each mixture component.

First pass

For our first pass of the model, we tried our best to implement it without tailoring too many of the model elements (standard deviations on normal priors, for example) to observed features of the data. This is a little different than the usual strategy of visually exploring data to determine how best to fit a model to said data.

We opted to do this to model how this analysis method might be used "in the wild." With most neuroimaging data, data are collected to fit a prescribed data structure (MRI image timeseries should be a certain structure, event markers for different sections of the timeseries should be labeled a certain way, etc), and researchers use a fairly stereotyped analysis pipeline that is theoretically robust to individual dataset-level variations. If we were to generalize the model we present here to future analyses, it would need to be completely generative, and thus not have parameters specified based on the data fed into it. If the generative model fits less well to the current dataset, we prefer that if the model can be generalized to future datasets without modification.

We opted to fit a mixture model with three ordered normal components. We constrained the first component to have a negative mean, the second component to have a mean of 0, and the third component to have a positive mean. We set the standard deviations of each of the mixture components to be equal, as we did not have a strong prior reason to believe the mixture components to have different spreads, and thus we wanted to minimize the number of parameters to be estimated.

We specified the η_{ROI} and $\eta_{subject}$ parameters similarly to the first, single normal model, but this time we specified η_{ROI} as a matrix with N rows, one for each ROI (as before), and with K columns, one for each mixture component (new). Each ROI has one η_{ROI_k} value, conditional on the ROI being drawn from mixture component k.

We added two new sets of parameters, β, for the means of each of the mixture components, and ϕ, for the latent variable indexing the probability that an observation would be drawn from a given mixture component.

Stan code

Here, we fix the middle mixture component to have a mean of 0 by passing 0 in as data.

See model1_1.stan for the relevant code.

```
fit1 <- stan("../stan/model1_1.stan",
                 data = betas %$%
                   list(beta_0 = 0,
                        N = nrow(.),
                        N_roi = max(roi_num),
                        N_subj = max(subject_num),
                        cope = cope_mean_scaled,
                        roi = roi_num,
                        subj = subject_num,
                        varcope = cope_sd_scaled),
                 control = list(max_treedepth = 15),
                 seed = 909)
```

Inspecting output

Now, to inspect the estimates for ϕ and β:

```
load("../../ignore/fit_model1.rda")

print(fit1, pars = c("phi", "beta"))
```

```
## Inference for Stan model: stan-1087918cf3752.
## 4 chains, each with iter=500; warmup=250; thin=1;
## post-warmup draws per chain=250, total post-warmup draws=1000.
##
##          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## phi[1]   0.00    0.00 0.00  0.00  0.00  0.00  0.00  0.01   917 1.00
## phi[2]   0.00    0.00 0.00  0.00  0.00  0.00  0.00  0.01  1245 1.00
```

```
## phi[3]   1.00    0.00 0.00  0.99  0.99  1.00  1.00  1.00   969 1.00
## beta[1] -2.16    0.09 2.63 -9.73 -3.25 -0.99 -0.30 -0.02   786 1.00
## beta[2]  0.00     NaN 0.00  0.00  0.00  0.00  0.00  0.00   NaN  NaN
## beta[3]  0.38    0.01 0.08  0.23  0.33  0.38  0.44  0.55   126 1.01
##
## Samples were drawn using NUTS(diag_e) at Mon Dec 10 00:14:02 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Inspecting the estimates for ϕ and β reveals that the mixture model does not actually add any new information. ϕ_3, the probability for the third (positive) mixture component, is functionally 1, while ϕ_1 and ϕ_2 are functionally 0. β_3 is estimated at about the mean of the observed data, while the other two mixture centers are estimated at 0–β_2 because we fixed it at 0, and β_1 presumably because it's centered at the mean value of the prior, and there are insufficient values in the observed data that would pull the posterior estimate below 0.

Attempts to refine model

We opted not to spend time running posterior predictive checks or other model evaluation on the first iteration of our mixture model, as we could tell immediately from checking the mixture parameter estimates that Stan had estimated our model as having essentially only one component.

We tried modifying the mixture model to remedy the non-mixture-estimation issue in the following ways. Stan code is printed for transparency, but model outputs are hidden for clarity, as we found that all of the modified models continued suffering from the problem of having one mixture component with $\phi \approx 1$ and the rest with $\phi \approx 0$.

Fewer mixture components

First, we cut the number of mixture components down to two, as we observed in the original single normal hierarchical model that there were hardly any ROIs for which η_{ROI} was "negative-activated", or much below 0. This way, we should only be estimating the "non-activated" and the "positive-activated" mixture components. We also stopped hard-coding the number of components, or the location of any of the components, in this model.

See model1_2.stan for the relevant code.

Once again, one of the mixture β values was estimated at the grand mean of the data with $\phi \approx 1$, and the other β was estimated at 0 (the center of the prior) with $\phi \approx 0$.

Fixed mixture centers

As a final attempt, we estimated a model where the mixture centers were not estimated as free parameters, but instead were fed into the model as data, so that we could manually specify the "expected" mixture centers.

See model1_3.stan for the relevant code.

We specified $\beta[2] = [0.3, 1]$, such that they should correspond to the "non-activated" and "positive-activated" mixture components respectively. The η_{ROI} values from the original single normal model showed a grand mean around 0.3, with a few ROIs having a higher η_{ROI} closer to 1, so we tried to equip the model to detect mixture components centered where the data might be centered.

Once more, $\phi_1 \approx 1$ for the "non-activated" component, while $\phi_2 \approx 0$ for the "positive-activated" component.

Model 1 summary

Ultimately, our mixture model extension failed to yield any insights beyond those of the single normal model. All models we attempted to fit estimated that 100% of our observations came from the same mixture component, suggesting that our data were not consistent with a mixture model after all.

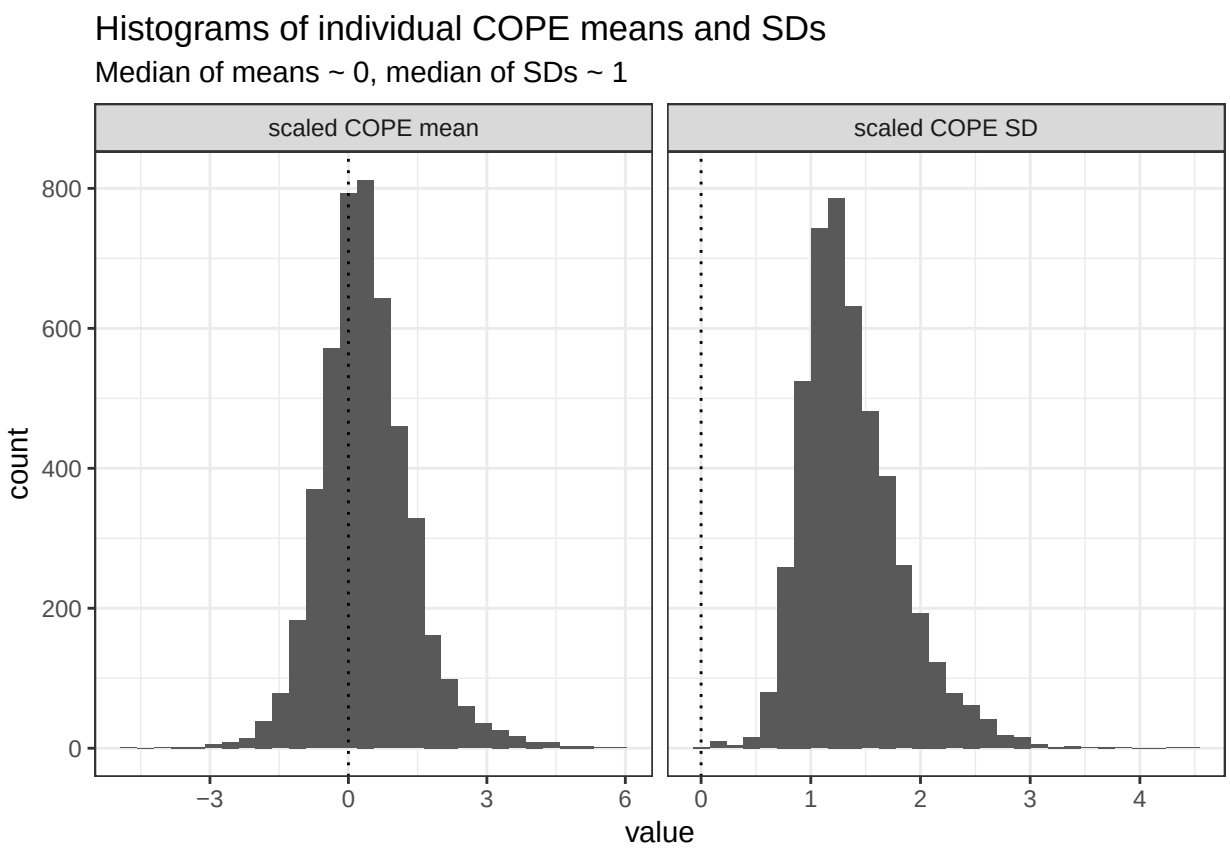
Discussion

Limitations of mixture modeling approach

We created our mixture model as independent of the actual features of the data as possible, in the hopes that if our hypothesis about the data-generating process were close to reality, we should see improved model fit by our a priori model specification alone. Ultimately, our data were not consistent enough with a mixture model for us to estimate any mixture-specific parameters.

Upon reflection and visual inspection, the data are not consistent with an obvious mixture process.

```
betas %>%
  select(`scaled COPE mean` = cope_mean_scaled, `scaled COPE SD` = cope_sd_scaled) %>%
  gather(key = "param", value = "value") %>%
  ggplot(aes(x = value)) +
  geom_histogram(bins = 30) +
  geom_vline(xintercept = 0, linetype = 3) +
  facet_wrap(~ param, scales = "free_x") +
  labs(title = "Histograms of individual COPE means and SDs",
       subtitle = "Median of means ~ 0, median of SDs ~ 1")
```

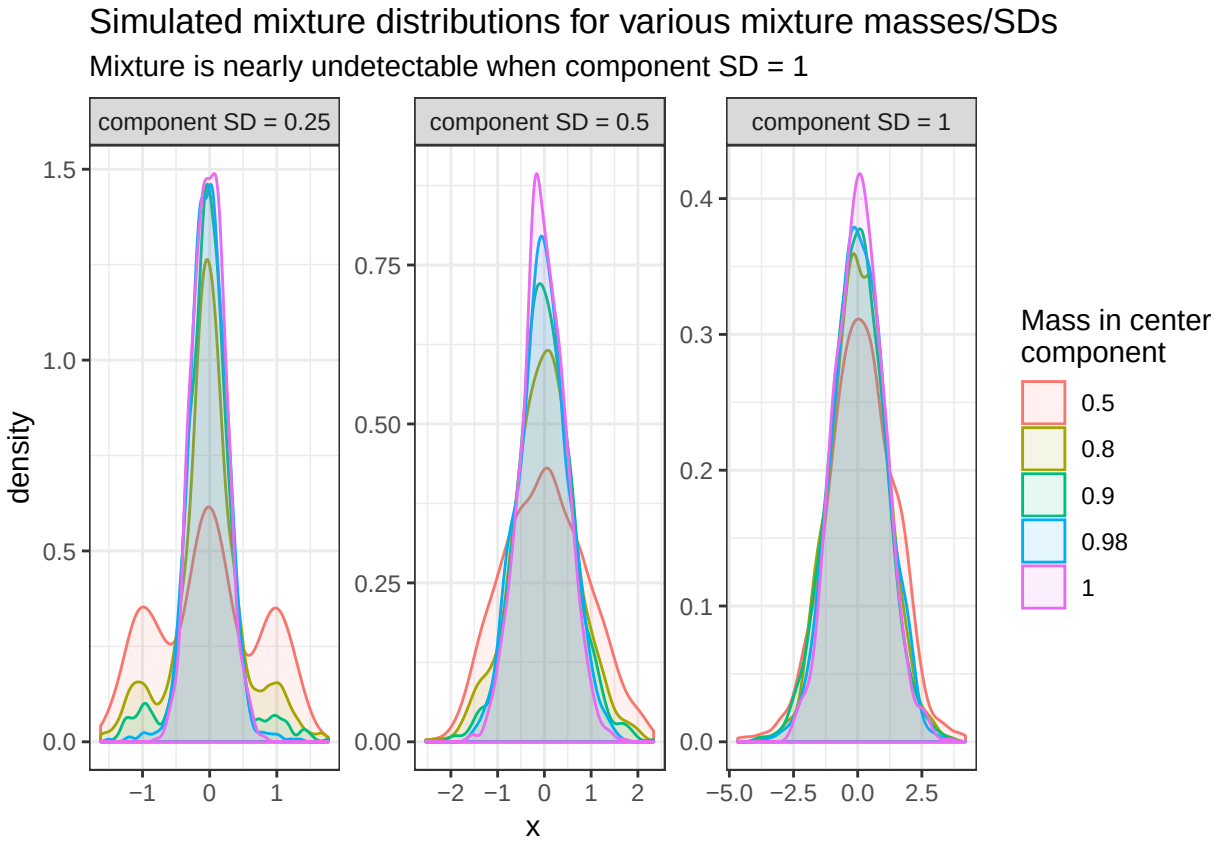


We can see now that even the estimates that might come from ROIs with $\eta_{ROI} \approx 1$ still have observation-level SDs that are probably close to or above 1. If the distance between mixture components is 1 standard deviation, *and* most observations come from one of the mixture components, the empirical distribution of the data probably looks very much like it comes from a single component model.

This still doesn't necessarily rule out that the data come from a mixture process, especially if that mixture process really does have one mixture component that generates most of the observations, with component SDs that are close to the distance between components. It may just be the case that with such mixture models, the data still look like they come from a single component model.

How much mass has to be in each component of a mixture process, and how narrow do the component SDs have to be, for observed data to look noticeably *unlike* a single component process? We can do a quick simulation to visualize. For simplicity, we'll assume a symmetrical three-component mixture process, with normal centers at -1, 0, and 1, and equal probability of data coming from either of the outer components.

```
crossing(id = 1:1000,
         sigma = c(0.25, 0.5, 1),
         center_prob = c(.5, .8, .9, .98, 1)) %>%
  mutate(side_prob = (1 - center_prob) / 2,
         probs = map2(center_prob, side_prob, ~c(.y, .x, .y)),
         mu = map_int(probs, ~sample(-1:1, size = 1, replace = TRUE,
                                     prob = .x)),
         x = rnorm(n(), mean = mu, sd = sigma),
         sigma = paste0("component SD = ", sigma)) %>%
  ggplot(aes(x = x, fill = factor(center_prob), color = factor(center_prob))) +
  # geom_histogram(position = "identity", alpha = 0.2) +
  geom_density(alpha = 0.1) +
  facet_wrap(~ sigma, scales = "free") +
  labs(title = "Simulated mixture distributions for various mixture masses/SDs",
       subtitle = "Mixture is nearly undetectable when component SD = 1",
       color = "Mass in center\ncomponent",
       fill = "Mass in center\ncomponent")
```



We can see from our simulation that when the mixture components are 1 SD wide (on the graph, "component SD = 1"), mixture data distributions are nearly indistinguishable from truly normal data (on the graph, "Mass in center component" = 1), even when only 50% of the mixture mass is in the center component. At smaller SDs, the mixture processes become more resolvable. Even then, when any more than 80% of the mixture mass is in the center component, the observed data are still basically unimodal, if a little thicker in the tails.

Our data are most comparable to the simulation with component SD = 1, as the unit-level SDs in our data are about the same magnitude as the expected difference between the "non-activated" and "positive-activated" mixture centers. Ultimately, our data *could* be consistent with a three-component mixture distribution whose mass is concentrated in the center component. However, that distribution is itself difficult to differentiate from a single normal distribution, so our data could be consistent with a single normal as well. We are thus unable to draw strong conclusions about how "mixed" our data really are.

Another noteworthy possibility, is that the assumption of qualitatively different groups of ROI activation in the brain is not warranted. Brain regions might plausibly come from one continuous distribution. Thus, the conceptual framework of fMRI analysis that sees brain regions coming from different distributions might have more to do with the NHST conceptualization, than with any real difference in activation. That is, since as researchers we were accustomed to looking for "significantly positive", "significantly negative" and "non-significant" regions, we adopted a theory of qualitatively different groups.

Future directions

While rudimentary, our Bayesian multilevel modeling approach to estimating whole-brain activity at the level of ROIs seem promising both with simulated and real data. We believe that this approach may offer an alternative that is more efficient and robust to type M/type S errors when estimating patterns of brain activity. However, several important considerations for application and extension of such models remain.

Voxelwise spatial autocorrelation

The mass univariate approach assumes to a degree that all voxels are independent units, but physiological patterns in the brain are known to be spatially autocorrelated. In addition, to decrease voxel-wise noise and increase power, it is standard in many fMRI analyses to employ 'spatial smoothing' across voxels; thus introducing even more spatial autocorrelation (Worsley, 2005). Although the ROI approach we have taken averages across spatial units comprised of many (hundreds or thousands) of voxels, as we plan to extend our model an additional level down to partially pool across all voxels, consideration of this spatial autocorrelation will be crucial. Indeed, even in our calculation of ROI-wise standard deviations, we have ignored this autocorrelation which may bias estimates. Future extensions of these models could make use of several methods for estimating and incorporating spatial autocorrelation, such as CAR models, or perhaps more simply, AR1 priors based on simple distance for voxelwise autocorrelation.

Repeated-measures fMRI data

Many fMRI studies examine brain activity in individuals across multiple scanning sessions. Our current model just estimates average activity across a sample of individuals assuming ever subject was scanned once, but we plan to adapt the model to accommodate repeated measurements within subjects to be able to estimate effects of interventions or longitudinal change. This would mean adding an additional hierarchical grouping factor in our model for, at the very least, allowing for intercepts to be estimated within-subject but across separate testing sessions.

Simulation-based power analysis and sample size calculations for fMRI studies

As we can use fully-specified versions of our model to simulate data directly from the priors, we can use MCMC iterations as 'datasets' in the future to conduct power analyses — that is, how large do effects in various ROIs have to be for us to reliably detect them with our model? Or, how large do within-subject or between-subjects differences in activity in an ROI have to be to detect them in a similar model with added regression parameters? Or, given an effect size, how many subjects in the dataset do we need to reliably find it? Such an application should be fairly straightforward in the future, as we used one iteration here to simulate data. We could either fix hyperparameters, or in a more Bayesian sense, center hyperparameter priors at our hypothesized effect size and draw samples from distributions.

Fully Bayesian fMRI inference from raw data to population estimates

While we have mostly focused on the population-level inferences in this project, there are also substantial aspects of fMRI processing pipeline for getting voxel-level estimates at the subject level that could benefit from Bayesian hierarchical modeling. In particular, because estimates at each voxel at the subject level are estimated predictors in a regression model based on each voxel's time series over the course of the scan, multilevel modeling of these time series data could help considerably in estimating activity more precisely for each voxel for each subject. This approach has some significant challenges in tractability however. Such a fully specified model would have at least several parameters for each voxel (and several hundred data points in the time series per voxel per subject), and with at least several hundred thousand parameters, this model would take an extremely large amount of computational power and time to fit. While this approach seems a valuable one, this presents a significant challenge, so we may have to devise methods of precomuting values for priors and hyperparameters for making this approach more computationally efficient. If possible though, this method of fully Bayesian inference could help substantially in regularizing noisy estimates at the subject level. We hope to continue pursing this possibility especially with the goal of improving the efficiency of fMRI analyses for answering questions in neuroscience and psychology.

References

Button, K. S., Ioannidis, J. P. A., Mokrysz, C., Nosek, B. A., Flint, J., Robinson, E. S. J., & Munafo, M. R. (2013). Power failure: why small sample size undermines the reliability of neuroscience. Nature Reviews Neuroscience, 14(5), 365???376. https://doi.org/10.1038/nrn3475

Chen, G., Xiao, Y., Taylor, P. A., Riggins, T., Geng, F., Redcay, E., & Cox, R. W. (2017). Handling Multiplicity in Neuroimaging through Bayesian Lenses with Hierarchical Modeling. BioRxiv, 238998. https://doi.org/10.1101/238998

Gee, D. G., Humphreys, K. L., Flannery, J., Goff, B., Telzer, E. H., Shapiro, M., Tottenham, N. (2013). A Developmental Shift from Positive to Negative Connectivity in Human Amygdala???Prefrontal Circuitry. Journal of Neuroscience, 33(10), 4584???4593. https://doi.org/10.1523/JNEUROSCI.3446-12.2013

Gelman, A., Hill, J., & Yajima, M. (2012). Why We (Usually) Don't Have to Worry About Multiple Comparisons. Journal of Research on Educational Effectiveness, 5(2), 189???211. https://doi.org/10.1080/19345747.2011.618213

Genovese, C. R., Lazar, N. A., & Nichols, T. (2002). Thresholding of Statistical Maps in Functional Neuroimaging Using the False Discovery Rate. NeuroImage, 15(4), 870???878. https://doi.org/10.1006/nimg.2001.1037

Geuter, S., Qi, G., Welsh, R. C., Wager, T. D., & Lindquist, M. A. (2018). Effect Size and Power in fMRI Group Analysis. BioRxiv, 295048. https://doi.org/10.1101/295048

Worsley, K. J. (2005). Spatial smoothing of autocorrelations to control the degrees of freedom in fMRI analysis. NeuroImage, 26(2), 635???641. https://doi.org/10.1016/j.neuroimage.2005.02.007