SMLP2018

Multinomial processing trees in Stan

Bruno Nicenboim

2018-09-12

Contents

Before we start the modeling of categorical responses, we will need the following packages, and it will also be useful to set a seed.

```r
set.seed(42)
library(dplyr)
library(extraDistr)
library(rstan)
# Save compiled models:
rstan_options(auto_write = TRUE)
# Parallelize the chains using all the cores:
options(mc.cores = parallel::detectCores())
# options(mc.cores = parallel::detectCores() - 1) # If you
# want to have an extra core free
library(bayesplot)
```

2 Modeling multiple categorical responses

One way to model categorical responses is using a multinomial or categorical distributions. The categorical responses could be Yes or No; blue, read or yellow; True, False, or I don't know; or more complicated categories, but crucially the response of each trial can be coded as only one category.

The multinomial and the categorical distribution represent two ways of parameterizing the same problem. The **multinomial distribution** is the generalization of the binomial distribution for more than two possible outcomes. For n independent trials each of which leads to a success for exactly one of k categories, with each category having a given fixed success probability, the multinomial distribution gives the probability of any particular combination of numbers of successes for the various categories. The **categorical distribution** is the generalization of the Bernoulli distribution for more than two possible outcomes, and it is the especial case of the multinomial distribution when we have only one trial.

2.1 A Practical example

Impaired picture naming (anomia) is common in most cases of aphasia and is assessed as part of most comprehensive aphasia test batteries, since picture naming accuracy is a relatively easily obtained, and reliable test score and the types of errors that are committed can provide useful information for diagnosis and specific targeting of these component processes in treatment.

In our fake experiment, we can categorize the responses to the task in the following way:

Category	Description	Example
Correct	The response matches the target.	cat
Neologism	The response is not a word, but it has a phonological relation to the target.	cag
Formal	The response is a word with only a phonological relation to the target.	hat
Mixed	The response is a word with both a semantic and phonological relation the target.	rat
NR	All other responses, including omissions, descriptions, non-nouns, etc.	–

2.1.1 The multinomial distribution

We are going to generate data assuming a multinomial distribution. The outcomes will be determined by a vector θ that indicates the probability of each outcome:

```r
true_theta <- tibble(theta_NR = 0.2, theta_Neologism = 0.1,
    theta_Formal = 0.2, theta_Mixed = 0.08, theta_Correct = 1 -
        (theta_NR + theta_Neologism + theta_Formal + theta_Mixed))
true_theta
```

```
## # A tibble: 1 x 5
##   theta_NR theta_Neologism theta_Formal theta_Mixed theta_Correct
##      <dbl>           <dbl>        <dbl>       <dbl>         <dbl>
## 1      0.2             0.1          0.2        0.08          0.42
```

```r
sum(true_theta)
```

```
## [1] 1
```

Given this θ we generate values assuming a multinomial distribution of responses in 100 trials:

```r
N_trials <- 100
(ans_mn <- rmultinom(1, N_trials, true_theta))
```

```
##                 [,1]
## theta_NR          26
## theta_Neologism   14
## theta_Formal      15
## theta_Mixed       10
## theta_Correct     35
```

We can build a model (`mn.stan`) in Stan assuming a multinomial distribution:

```stan
data {
  int<lower=1> N_trials;
  int<lower=0,upper=N_trials> ans[5];
}
parameters {
  simplex[5] theta;
}
model {
  theta ~ beta(2, 2);
  ans ~ multinomial(theta);
}
generated quantities{
    int pred_ans[5];
  pred_ans = multinomial_rng(theta, 5);
}
```

And we fit the model:

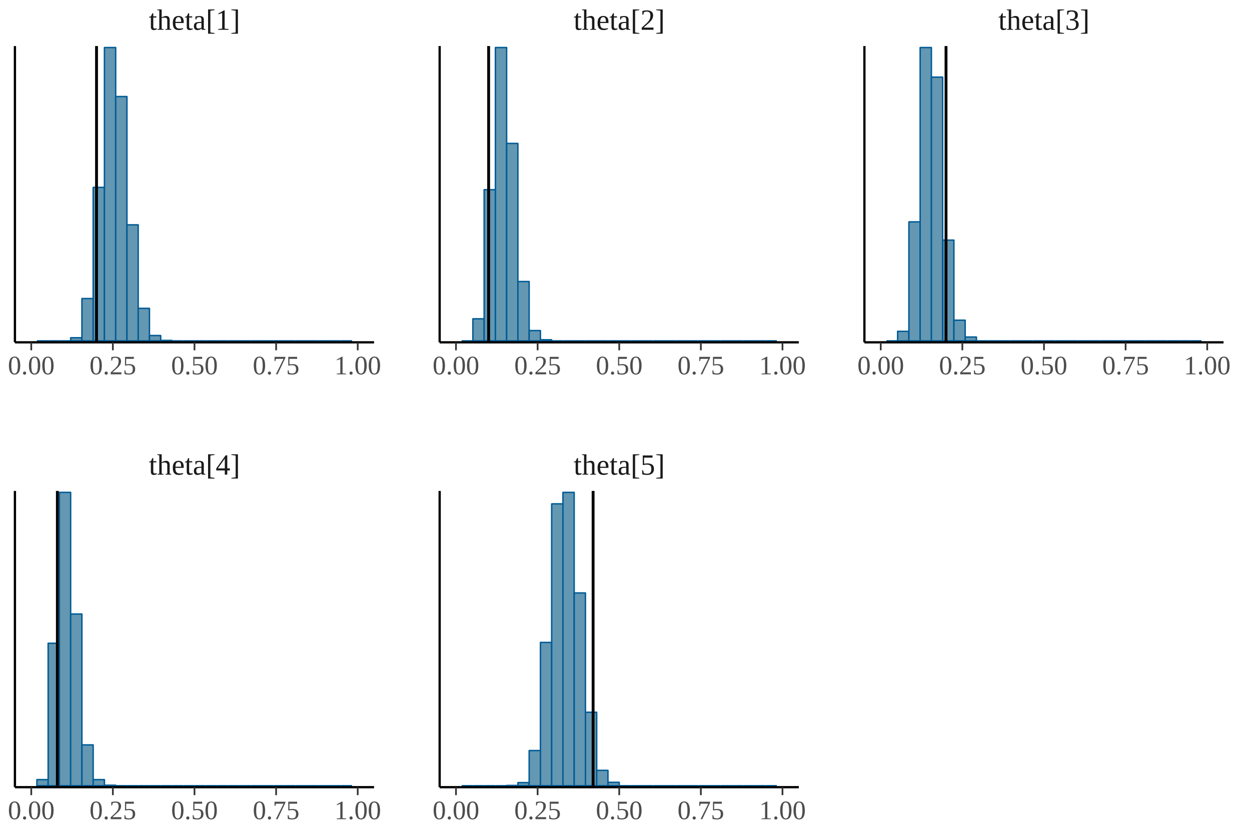
```r
# Create a list: c(ans_mn) makes a vector out of the
# 'tibble'
data_mn <- list(N_trials = N_trials, ans = c(ans_mn))

# We fit the model with the default values of number of
# chains and iterations: chains = 4, iter = 2000
fit_mn <- stan(file = "mn.stan", data = data_mn)
```

We can inspect that the posterior of θ is close to the true value that we selected.

```r
print(fit_mn, pars = c("theta"), probs = c(0.0275, 0.5, 0.975))
```

```
## Inference for Stan model: mn.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd 2.75%  50% 97.5% n_eff Rhat
## theta[1] 0.25       0 0.04  0.18 0.25  0.34  4000    1
## theta[2] 0.15       0 0.03  0.08 0.14  0.21  4000    1
## theta[3] 0.15       0 0.03  0.09 0.15  0.23  4000    1
## theta[4] 0.11       0 0.03  0.06 0.11  0.18  3869    1
## theta[5] 0.34       0 0.04  0.25 0.33  0.42  4000    1
##
## Samples were drawn using NUTS(diag_e) at Wed Sep 12 15:15:14 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).
```

```r
mcmc_hist(as.array(fit_mn), regex_pars = "theta") + geom_vline(data = tibble(Parameter
    1:5, "]"), value = unlist(true_theta)), aes(xintercept = value)) +
    scale_x_continuous(limits = c(0, 1))
```



2.1.2 The categorical distribution

Using the same information, we can model one response at a time, instead of aggregating them. We'll see that this gives us more flexibility to define what happens at every trial (cat.stan).

```
data {
  int<lower=1> N_trials;
  int<lower=1,upper=5> w_ans[N_trials];
}
parameters {
  simplex[5] theta;
}
model {
  theta ~ beta(2, 2);
  for(n in 1:N_trials)
    w_ans[n] ~ categorical(theta);
}
```

```
generated quantities{
  int pred_w_ans[N_trials];
  for(n in 1:N_trials)
    pred_w_ans[n] = categorical_rng(theta);
}
```

Given the same θ we now generate 100 individual trials:

```
ans_cat <- rmultinom(N_trials, 1, true_theta)
head(ans_cat[, 1:20])

# We need to change the answer to make it compatible to what
# Stan expects (a response between 1-5 that corresponds to
# our categories).

data_cat <- list(N_trials = N_trials, w_ans = c(t(ans_cat) %*%
    1:5))  # This makes matrix into a vector of responses between 1 and 5
```

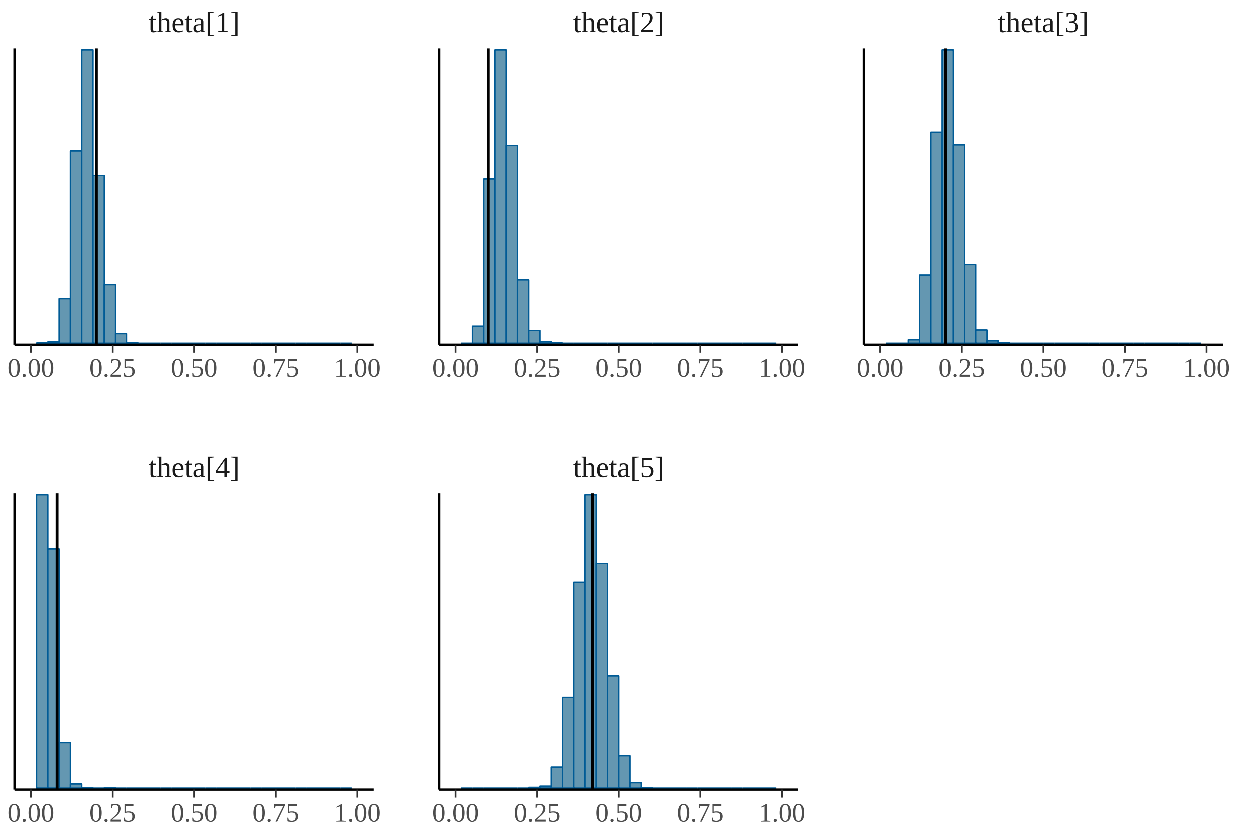
We should get roughly the same θ as before:

```
fit_cat <- stan(file = "cat.stan", data = data_cat)
```

```
print(fit_cat, pars = c("theta"), probs = c(0.0275, 0.5, 0.975))
```

```
## Inference for Stan model: cat.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd 2.75%  50% 97.5% n_eff Rhat
## theta[1] 0.17       0 0.04  0.11 0.17  0.25  4000    1
## theta[2] 0.15       0 0.03  0.09 0.14  0.22  4000    1
## theta[3] 0.21       0 0.04  0.14 0.21  0.29  4000    1
## theta[4] 0.05       0 0.02  0.02 0.05  0.10  4000    1
## theta[5] 0.42       0 0.05  0.33 0.42  0.51  4000    1
##
## Samples were drawn using NUTS(diag_e) at Wed Sep 12 15:15:20 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).
```

```
mcmc_hist(as.array(fit_cat), regex_pars = "theta") + geom_vline(data = tibble(Parameter
    1:5, "]"), value = unlist(true_theta)), aes(xintercept = value)) +
    scale_x_continuous(limits = c(0, 1))
```



2.2 Exercises

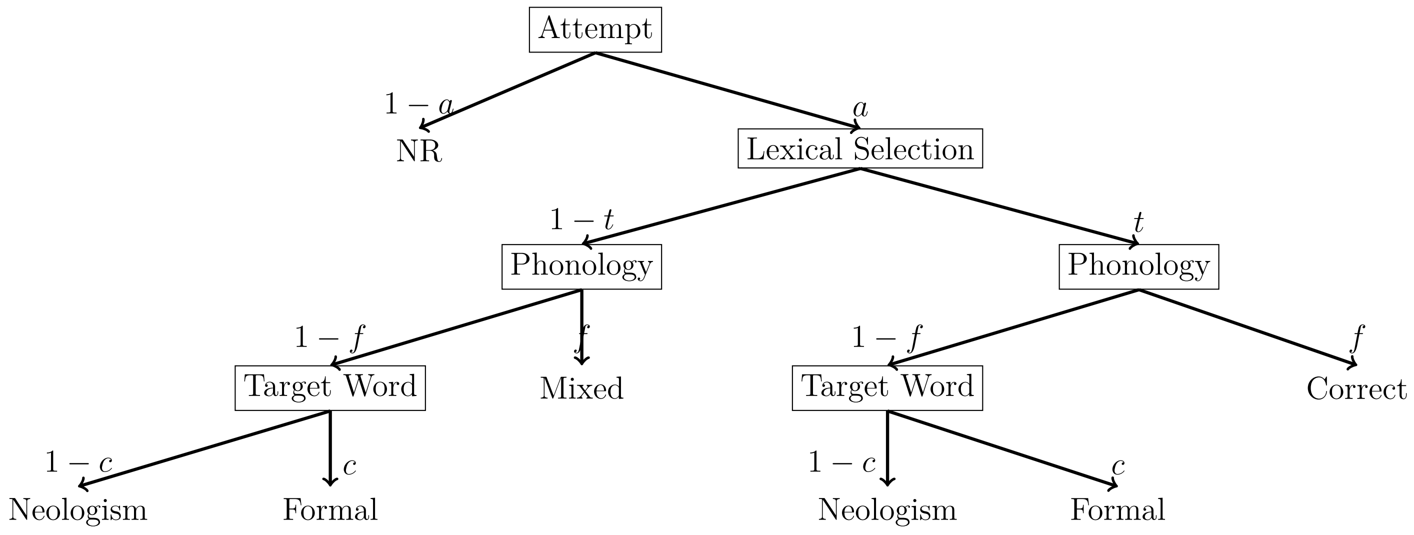
1. Try the models with different number of trials.
2. Assume that you have more information about the the probability of making a correct response. Say that your domain knowledge tells you that it will be around 60%. Encode this information with different degrees of certainty.
3. What is the difference between two presented parameterization of the same underlying model?
4. What can we learn about impaired picture naming?

3 Multinomial processing tree (MPT) models

Multinomial processing tree (MPT) model is a method that estimates latent variables that have a psychological interpretation given categorical data (see a review in Batchelder and Riefer 1999), in other words, a MPT model is *one* way to model categorical responses following a multinomial or categorical distribution. MPT models assume that the observed response categories result from a sequences of underlying cognitive events which are represented as a tree.

3.1 An application of MPTs in modeling picture naming abilities in aphasia

Walker, Hickok, and Fridriksson (2018) created an MPT model that specifies a set of possible internal errors that lead to the various possible response types during a picture naming trial for aphasic patients. Here we'll explore a simplification of the original model.



The parameters of the MPT model represent the following:

Param.	Description
a	Probability of initiating an attempt
t	Probability of selecting a target word over competitors
f	Probability of retrieving correct phonemes
c	Probability of a phoneme change in the target word creating a real word

3.1.1 Estimation of the probabilities

Navigating through the branches of the MPT, we can estimate the probabilities of the the four responses (the categorical outcomes), based on the underlying parameters that we assumed:

- $Pr(NR|a,t,f,c) = 1 - a$

- $Pr(Neologism|a,t,f,c) = a \cdot (1-t) \cdot (1-f) \cdot (1-c) + a \cdot t \cdot (1-f) \cdot (1-c)$
- $Pr(Formal|a,t,f,c) = a \cdot (1-t) \cdot (1-f) \cdot c + a \cdot t \cdot (1-f) \cdot c$
- $Pr(Mixed|a,t,f,c) = a \cdot (1-t) \cdot f$
- $Pr(Correct|a,t,f,c) = a \cdot t \cdot f$

Given that $Pr(NR|a,t,f,c)+Pr(Neologism|a,t,f,c)+Pr(Formal|a,t,f,c)+Pr(Mixed|a,t,f,c)+Pr(Correct|a,t,f,c) = 1$ we don't need to characterize every outcome, and we can always estimate one response as $1 - other\ responses$.

3.1.2 Fake data generation of the possible outcomes

Let's simulate 200 trials assuming homogeneity of items and participants:

```
# Probabilities of different answers
Pr_NR <- function(a, t, f, c) {
    1 - a
}
Pr_Neologism <- function(a, t, f, c) {
    a * (1 - t) * (1 - f) * (1 - c) + a * t * (1 - f) * (1 -
        c)
}
Pr_Formal <- function(a, t, f, c) {
    a * (1 - t) * (1 - f) * c + a * t * (1 - f) * c
}
Pr_Mixed <- function(a, t, f, c) {
    a * (1 - t) * f
}
Pr_Correct <- function(a, t, f, c) {
    a * t * f
}

# Fake true underlying values
a_true <- 0.75
t_true <- 0.9
f_true <- 0.8
c_true <- 0.1
```

```r
# Prob of the different answers:
(theta_NR <- Pr_NR(a_true, t_true, f_true, c_true))
```

```
## [1] 0.25
```

```r
(theta_Neologism <- Pr_Neologism(a_true, t_true, f_true, c_true))
```

```
## [1] 0.135
```

```r
(theta_Formal <- Pr_Formal(a_true, t_true, f_true, c_true))
```

```
## [1] 0.015
```

```r
(theta_Mixed <- Pr_Mixed(a_true, t_true, f_true, c_true))
```

```
## [1] 0.06
```

```r
(theta_Correct <- Pr_Correct(a_true, t_true, f_true, c_true))
```

```
## [1] 0.54
```

```r
N_trials <- 200

ans <- rmultinom(1, N_trials, c(theta_NR, theta_Neologism, theta_Formal,
    theta_Mixed, theta_Correct))
```

3.1.3 A simple MPT model in Stan

It is straightforward to model this in Stan (see `aphasia_sMPT.stan`), the probabilities of the different categories go into the `transformed parameters` section. And the data is modeled as a multinomial distribution; if priors are not specified, then a beta distribution with $a = 1$ and $b = 1$ is assumed for the parameters a, t, f, and c.

```stan
data {
  int<lower=1> N_trials;
  int<lower=0,upper=N_trials> ans[5];
}
parameters {
  real<lower=0,upper=1> a;
  real<lower=0,upper=1> t;
  real<lower=0,upper=1> f;
  real<lower=0,upper=1> c;
}
transformed parameters {
  simplex[5] theta;
  theta[1] = 1 - a; //Pr_NR
 theta[2] = a * (1 - t) * (1 - f) * (1 - c) + a * t * (1 - f) * (1 - c); //Pr_Neologism
  theta[3] = a * (1 - t) * (1 - f) * c +  a * t * (1 - f) * c;  //Pr_Formal
  theta[4] = a * (1 - t) * f; //Pr_Mixed
  theta[5] = a * t * f; //Pr_Correct
}
model {
  a ~ beta(2, 2);
  t ~ beta(2, 2);
  f ~ beta(2, 2);
  c ~ beta(2, 2);
  ans ~ multinomial(theta);
}
generated quantities{
    int pred_ans[5];
  pred_ans = multinomial_rng(theta, 5);
}
```

We fit the model

```
data_sMPT <- list(N_trials = N_trials, ans = c(ans))
fit_sMPT <- stan(file = "aphasia_sMPT.stan", data = data_sMPT)
```

```
## Warning: There were 1 transitions after warmup that exceeded the maximum treedepth. Incre
## http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

We can see a summary of the posterior of the parameter of interest:

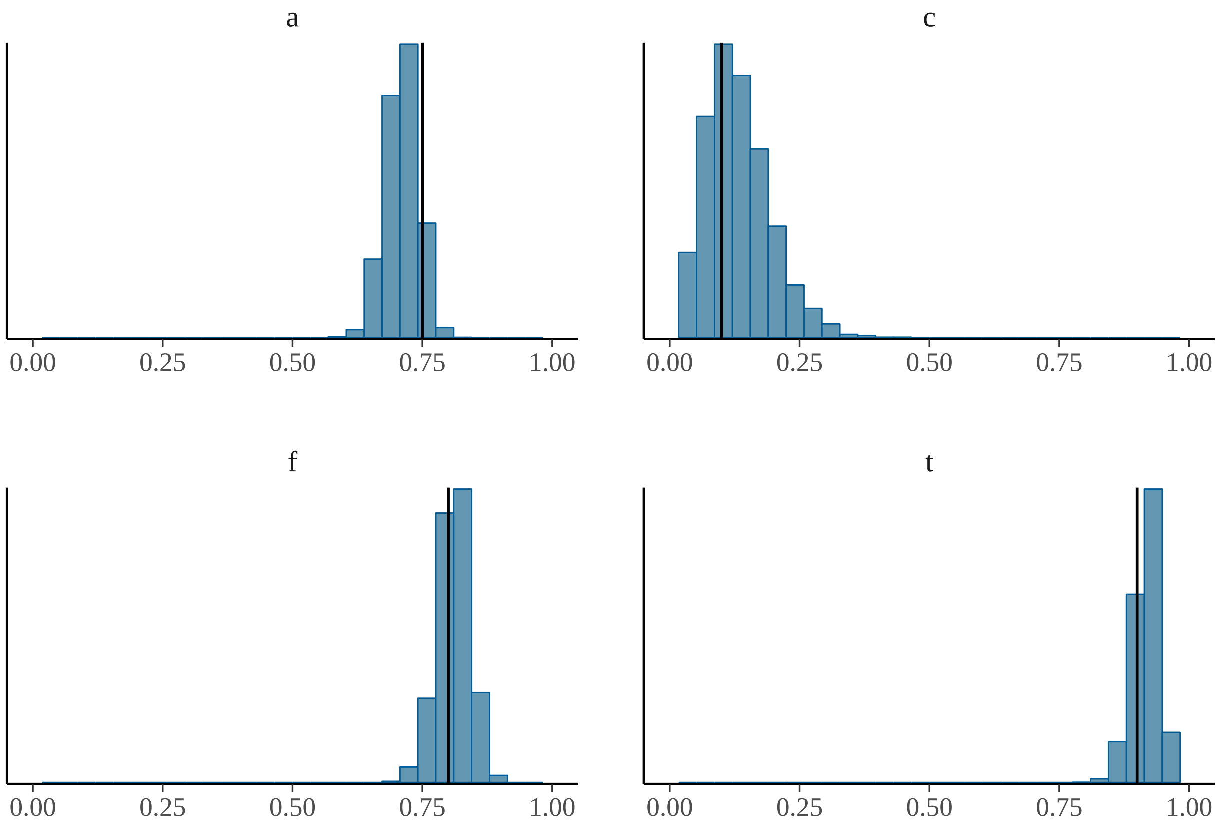
```
print(fit_sMPT, pars = c("a", "t", "f", "c"), probs = c(0.0275,
    0.5, 0.975))
```

```
## Inference for Stan model: aphasia_sMPT.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
```

```
##
##   mean se_mean   sd 2.75%  50% 97.5% n_eff Rhat
## a 0.71       0 0.03  0.65 0.71  0.77  4000    1
## t 0.92       0 0.02  0.86 0.92  0.96  4000    1
## f 0.81       0 0.03  0.75 0.81  0.87  4000    1
## c 0.13       0 0.06  0.04 0.12  0.28  4000    1
##
## Samples were drawn using NUTS(diag_e) at Wed Sep 12 15:15:25 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).
```

We can evaluate if our model is not doing something crazy, since we know the true values of the parameters:

```
mcmc_hist(as.array(fit_sMPT), pars = c("a", "t", "f", "c")) +
    geom_vline(data = tibble(Parameter = c("a", "t", "f", "c"),
        value = c(a_true, t_true, f_true, c_true)), aes(xintercept = value)) +
    scale_x_continuous(limits = c(0, 1))
```



We can also recover the posterior of θ, but it is not very interesting:

```
print(fit_sMPT, pars = c("theta"), probs = c(0.0275, 0.5, 0.975))
```

```
## Inference for Stan model: aphasia_sMPT.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd 2.75%  50% 97.5% n_eff Rhat
## theta[1] 0.29       0 0.03  0.23 0.29  0.35  4000    1
## theta[2] 0.12       0 0.02  0.08 0.12  0.16  4000    1
## theta[3] 0.02       0 0.01  0.00 0.02  0.04  4000    1
## theta[4] 0.05       0 0.01  0.02 0.05  0.08  4000    1
## theta[5] 0.53       0 0.03  0.46 0.53  0.59  4000    1
##
## Samples were drawn using NUTS(diag_e) at Wed Sep 12 15:15:25 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).
```

3.1.4 Exercises

1. How does the estimation of the parameter change if we only have 10 trials?
2. We may have some information about the parameter a. We might assume that participants should be motivated to try to do the task. How should we encode this with different degrees of certainty (i.e., in different models with different priors)?
3. Try the previous model with 10 and with 100 trials. What can you conclude?
4. Is it possible to falsify this model? Can you come up with some data that we cannot predict?

3.1.5 A slightly more complex Stan model–adding a regression to one of the parameters.

The use of aggregated data relies on the assumption that the estimated parameters do not vary (too much at least) between subjects and items. If this assumption is violated, the analysis of aggregated data may lead to erroneous conclusions: reliance on aggregated data in the presence of parameter heterogeneity may lead to biased parameter estimates and the underestimation of credible intervals.

For example, even if we knew that f is affected by the phonological complexity of the word (e.g., cat is easier to produce than umbrella), the previous model does not have a way to include that information.

We can now generate fake data where we have measured the `complexity` of the items (which is scaled and centered). We will assume a regression that will look like $f' = f_\alpha + complexity \cdot f_\beta$. One important detail is that `f` is a probability and needs to be bounded between 0 and 1. While we will calculate f_α and f_β in an unconstrained space (they can be any real number), we can bound f' to a be a probability by applying the logistic function (or $logit^{-1}$) to f':

```r
N_trials <- 1000
complexity <- rnorm(N_trials, 0, 2)

f_alpha <- 0.3
f_beta <- 0.05

# This is where the magic happens:
f_true <- plogis(f_alpha + complexity * f_beta)
```

This change in our assumptions entails that the probability of each response changes with the item. And `theta` is now a matrix (this is in R, in Stan, we'll code it as an array of simplexes).[1]

```r
theta_NR_v <- rep(Pr_NR(a_true, t_true, f_true, c_true), N_trials)
theta_Neologism_v <- Pr_Neologism(a_true, t_true, f_true, c_true)
theta_Formal_v <- Pr_Formal(a_true, t_true, f_true, c_true)
theta_Mixed_v <- Pr_Mixed(a_true, t_true, f_true, c_true)
theta_Correct_v <- Pr_Correct(a_true, t_true, f_true, c_true)

theta_item <- matrix(c(theta_NR_v, theta_Neologism_v, theta_Formal_v,
    theta_Mixed_v, theta_Correct_v), ncol = 5)

fake_data_cx <- tibble(item = 1:N_trials, complexity = complexity,
    w_ans = c(rmnom(N_trials, 1, theta_item) %*% 1:5))
```

The following model `aphasia_sMPT_cat.stan` is essentially doing the same as the previous model but instead of fitting a multinomial to the summary of all the trials, it is fitting a

[1]Notice below that if the equations depended on `f`, which is a vector now, the outcome is automatically a vector. But this is not the case for `theta_NR_v`, and thus we need to repeat the value.

categorical distribution to each individual trial. (This is parallel to the difference between Bernoulli and binomial distribution). It is not the appropriate model for the generative process that we are assuming in this section, but it is a good start.

```
data {
  int<lower=1> N_trials;
  int<lower=1,upper=5> w_ans[N_trials];
}
parameters {
  real<lower=0,upper=1> a;
  real<lower=0,upper=1> t;
  real<lower=0,upper=1> f;
  real<lower=0,upper=1> c;
}
transformed parameters {
  simplex[5] theta[N_trials];

  for(n in 1:N_trials){
    theta[n, 1] = 1 - a; //Pr_NR
   theta[n, 2] = a * (1 - t) * (1 - f) * (1 - c) + a * t * (1 - f) * (1 - c); //Pr_Neologism
    theta[n, 3] = a * (1 - t) * (1 - f) * c +  a * t * (1 - f) * c;  //Pr_Formal
    theta[n, 4] = a * (1 - t) * f; //Pr_Mixed
    theta[n, 5] = a * t * f; //Pr_Correct
  }
}
model {
  a ~ beta(2, 2);
  t ~ beta(2, 2);
  f ~ beta(2, 2);
  c ~ beta(2, 2);

  for(n in 1:N_trials)
    w_ans[n] ~ categorical(theta[n]);
}
generated quantities{
    int pred_w_ans[N_trials];
  for(n in 1:N_trials)
```

```
    pred_w_ans[n] = categorical_rng(theta[n]);
}
```

3.1.6 Exercises

1. Modify the previous model so that we can include data about `complexity`, the new parameters, `f_alpha` and `f_beta`, and the auxiliary or transformed parameter `f` that will now vary depending on the complexity of the item. Fit it to the data.
2. Plot the posteriors of all the relevant parameters. What would you say about the effect of complexity?

3.1.7 A hierarchical MPT in Stan

The previous model doesn't take into account that subjects might vary and that items might vary beyond what is acounted by complexity. Let's focus first in taking into account the differences between subject.

Different subjects might be differently motivated to the task, and this can be accounted by adding a hierarchical structure to the parameter `a`. We'll generate subjects and items, adding this assumption to the new fake data.

```
N_item <- 20
N_subj <- 30
N_trials <- N_item * N_subj
subject <- rep(1:N_subj, each = N_item)
item <- rep(1:N_item, time = N_subj)
complexity <- rnorm(N_trials, 0, 2)

exp_fake <- tibble(subject = subject, item = item, complexity = complexity)

sigma_a_alpha_subject <- 1.1
a_alpha_subject_tilde <- rnorm(N_subj, 0, sigma_a_alpha_subject)
a_true_h <- plogis(qlogis(a_true) + sigma_a_alpha_subject *
    a_alpha_subject_tilde[subject])

f_true <- plogis(f_alpha + complexity * f_beta)

theta_NR_v_h <- Pr_NR(a_true_h, t_true, f_true, c_true)
```

```
theta_Neologism_v_h <- Pr_Neologism(a_true_h, t_true, f_true,
    c_true)
theta_Formal_v_h <- Pr_Formal(a_true_h, t_true, f_true, c_true)
theta_Mixed_v_h <- Pr_Mixed(a_true_h, t_true, f_true, c_true)
theta_Correct_v_h <- Pr_Correct(a_true_h, t_true, f_true, c_true)

theta_h <- matrix(c(theta_NR_v_h, theta_Neologism_v_h, theta_Formal_v_h,
    theta_Mixed_v_h, theta_Correct_v_h), ncol = 5)

fake_data_h <- mutate(exp_fake, w_ans = c(rmnom(N_trials, 1,
    theta_h) %*% 1:5))
```

The appropriate model `aphasia_MPT_h.stan` will look like this:

```
data {
  int<lower = 1> N_trials;
  int<lower = 1,upper = 5> w_ans[N_trials];
  real complexity[N_trials];
  int<lower = 1> N_subject;
  int<lower = 1, upper = N_subject> subject[N_trials];
}
parameters {
  real<lower = 0, upper = 1> t;
  real<lower = 0, upper = 1> c;
  real a_alpha;
  real<lower = 0> sigma_a_alpha_subject;
  vector[N_subject] a_alpha_subject_tilde;
  real f_alpha;
  real f_beta;
}
transformed parameters {
  simplex[5] theta[N_trials];
  for (n in 1:N_trials){
   real a = inv_logit(a_alpha + sigma_a_alpha_subject * a_alpha_subject_tilde[subject[n]])
    real f = inv_logit(f_alpha + complexity[n] * f_beta);
    theta[n, 1] = 1 - a; //Pr_NR
   theta[n, 2] = a * (1 - t) * (1 - f) * (1 - c) + a * t * (1 - f) * (1 - c); //Pr_Neologism
```

```
    theta[n, 3] = a * (1 - t) * (1 - f) * c +  a * t * (1 - f) * c;  //Pr_Formal
    theta[n, 4] = a * (1 - t) * f; //Pr_Mixed
    theta[n, 5] = a * t * f; //Pr_Correct
  }
}
model {
  t ~ beta(2, 2);
  c ~ beta(2, 2);
  f_alpha ~ normal(0, 2);
  f_beta ~ normal(0, 2);
  a_alpha ~ normal(0, 2);
  a_alpha_subject_tilde ~ normal(0, 1);

  for(n in 1:N_trials)
    w_ans[n] ~ categorical(theta[n]);
}
generated quantities{
  int<lower = 1, upper = 5> pred_w_ans[N_trials];
  for(n in 1:N_trials)
    pred_w_ans[n] = categorical_rng(theta[n]);
}
```

And we can now fit it to our fake data.

```r
fake_list_h <- list(N_trials = nrow(fake_data_h), w_ans = fake_data_h$w_ans,
    N_subject = max(fake_data_h$subject), subject = fake_data_h$subject,
    complexity = fake_data_h$complexity)

fit_h <- stan(file = "aphasia_MPT_h.stan", data = fake_list_h,
    control = list(adapt_delta = 0.9))
```

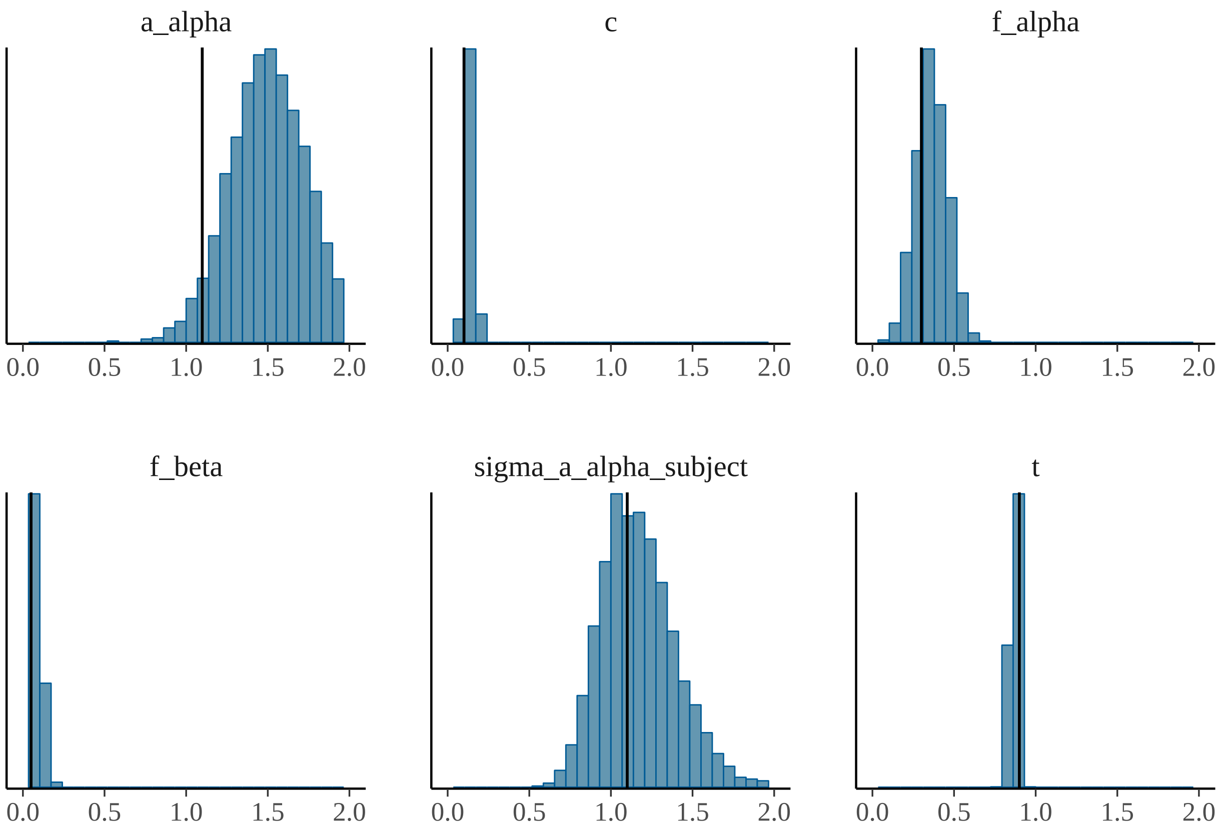
We can see a summary of the posterior

```r
print(fit_h, pars = c("t", "c", "sigma_a_alpha_subject", "a_alpha",
    "f_alpha", "f_beta"), probs = c(0.0275, 0.5, 0.975))
```

```
## Inference for Stan model: aphasia_MPT_h.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
```

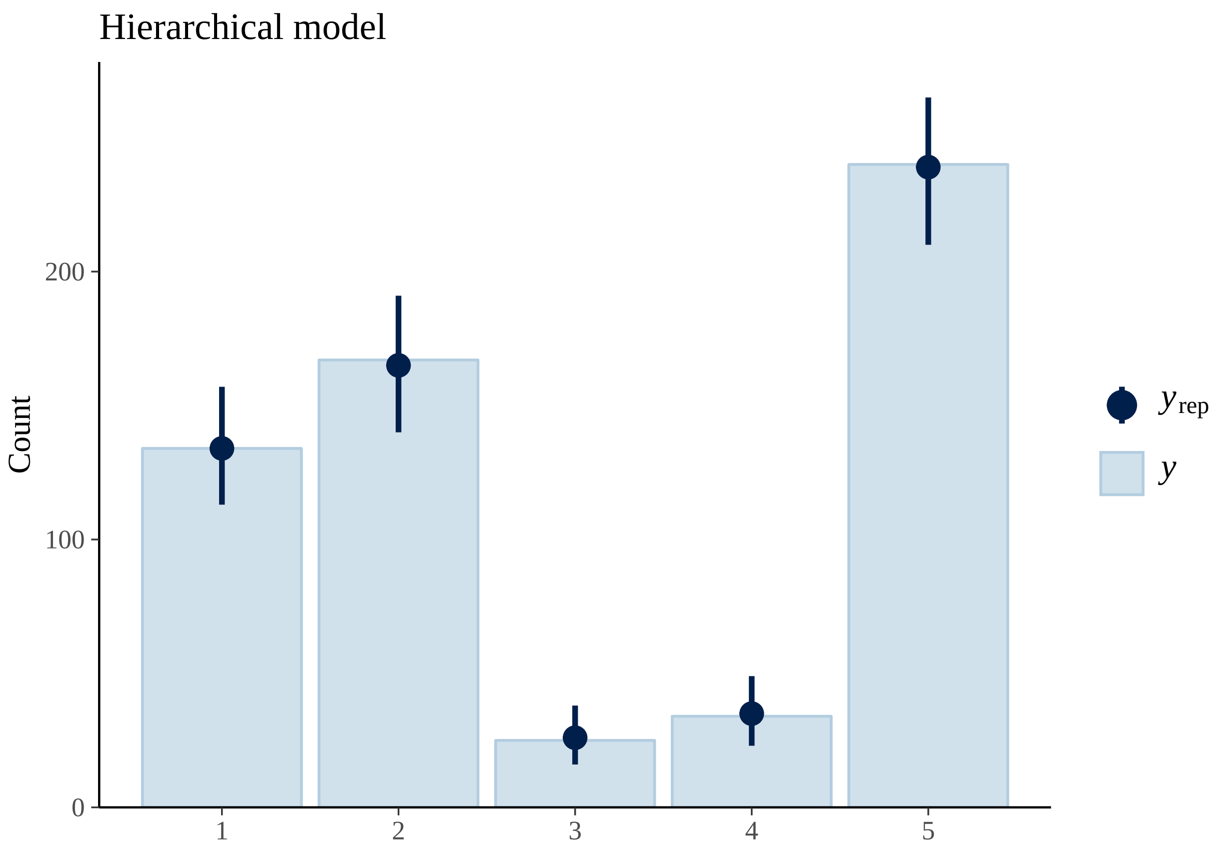
```
##
##                        mean se_mean   sd 2.75%  50% 97.5% n_eff Rhat
## t                      0.87    0.00 0.02  0.83 0.87  0.91  4000    1
## c                      0.14    0.00 0.02  0.09 0.14  0.19  4000    1
## sigma_a_alpha_subject 1.17    0.01 0.23  0.79 1.15  1.67  1426    1
## a_alpha                1.52    0.01 0.26  1.05 1.51  2.06  1501    1
## f_alpha                0.36    0.00 0.10  0.18 0.36  0.55  4000    1
## f_beta                 0.06    0.00 0.05 -0.03 0.06  0.16  4000    1
##
## Samples were drawn using NUTS(diag_e) at Wed Sep 12 15:17:14 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).
```

```
mcmc_hist(as.array(fit_h), pars = c("sigma_a_alpha_subject",
    "a_alpha", "t", "f_alpha", "f_beta", "c")) + geom_vline(data = tibble(Parameter = c(
    "a_alpha", "t", "f_alpha", "f_beta", "c"), value = c(sigma_a_alpha_subject,
    qlogis(a_true), t_true, f_alpha, f_beta, c_true)), aes(xintercept = value)) +
    scale_x_continuous(limits = c(0, 2))
```



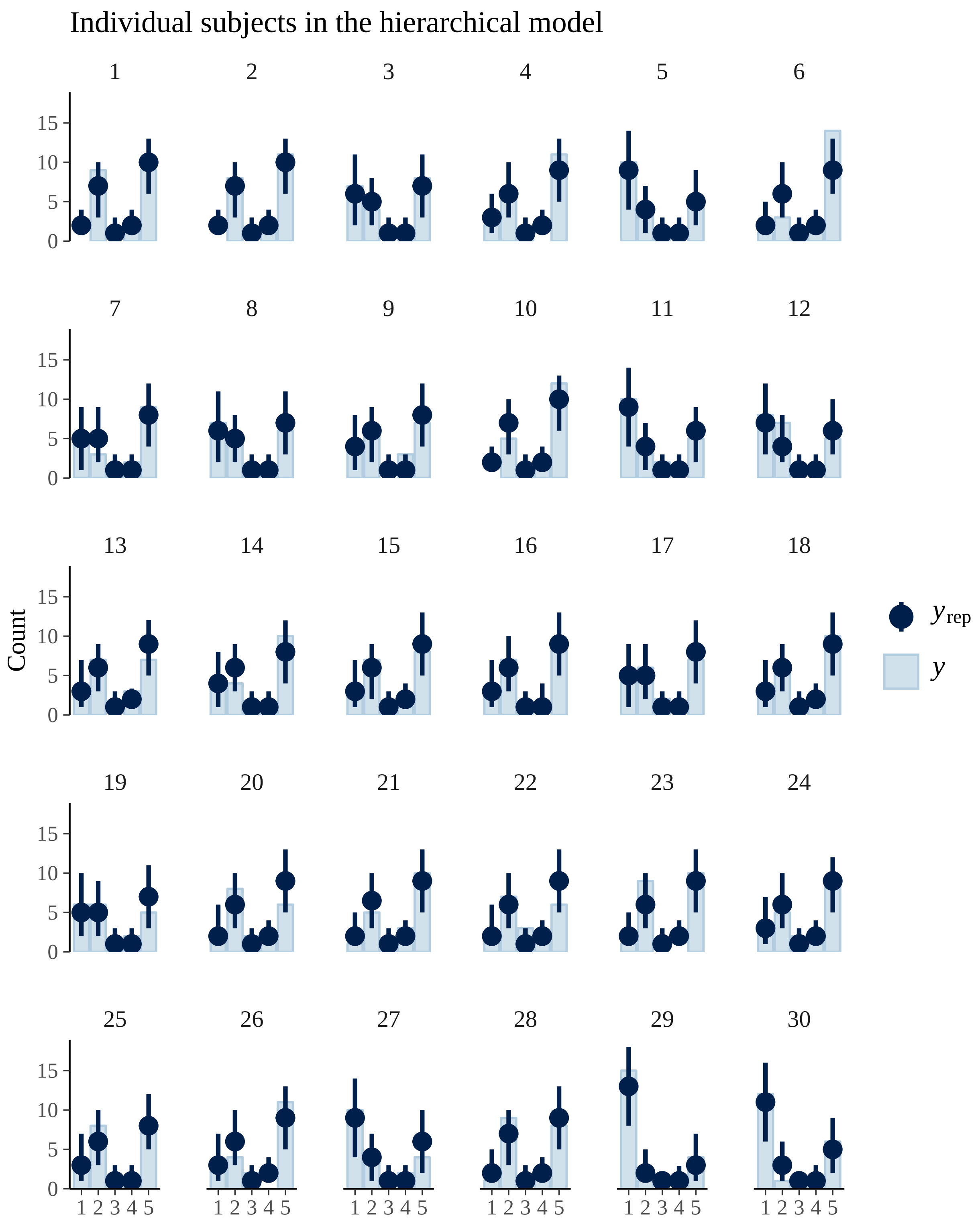
If we did everything fine, we should be able to generate data based on our estimates that looks like our "fake data":

```r
gen_data <- rstan::extract(fit_h)$pred_w_ans
ppc_bars(fake_list_h$w_ans, gen_data) + ggtitle("Hierarchical model")
```



And we can look at our individual subjects:

```r
ppc_bars_grouped(fake_list_h$w_ans, gen_data, group = subject) +
    ggtitle("Individual subjects in the hierarchical model")
```

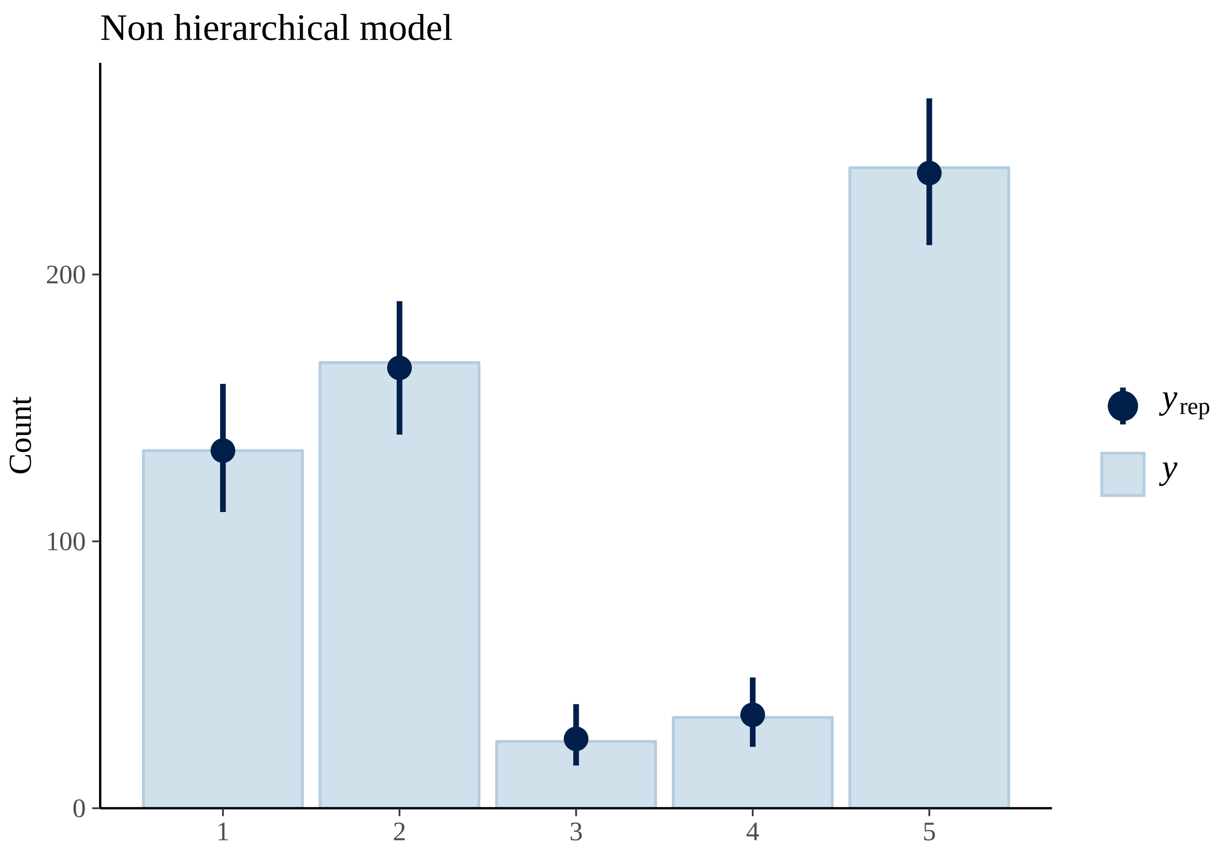


But what about the first *non-hierarchical* MPT?:

```
fit_sh <- stan(file = "aphasia_sMPT_cat.stan", data = fake_list_h)
```

The aggregated data looks great

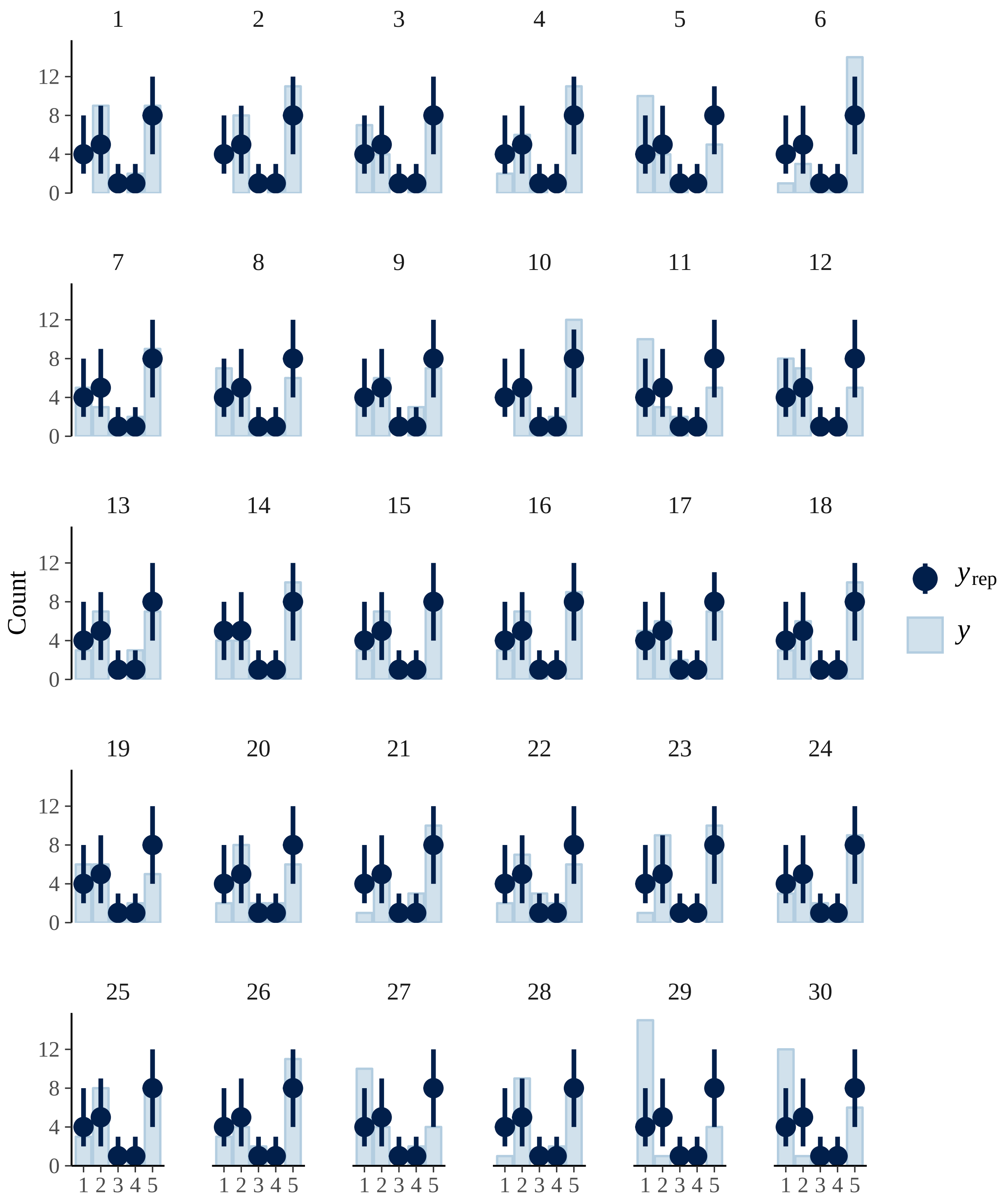
```r
gen_data_sMPT <- rstan::extract(fit_sh)$pred_w_ans
ppc_bars(fake_list_h$w_ans, gen_data_sMPT) + ggtitle("Non hierarchical model")
```



However, the fit to individual subjects looks less good:

```r
ppc_bars_grouped(fake_list_h$w_ans, gen_data_sMPT, group = subject) +
    ggtitle("Individual subjects in the non hierarchical model")
```

Individual subjects in the non hierarchical model



3.1.8 Exercises

1. Bin the complexity in a number of groups, and compare the first non-hierarchical model (which also doesn't account for complexity) with our hierarchical model using posterior predictive checks.
2. How would you represent the different `a` of each subject. Plot them. Hint: use `a_alpha_subject_tilde` in the generated quantities.
3. Other parameters might also be affected by the subject. How would the data change? (Create new fake data) How would the model change? Fit it to the new fake data.
4. Parameters might also be affected by items. How would the data change? (Create new fake data) How would the model change? Fit it to the new fake data.

3.2 Recommended readings about Bayesian MPT:

- Matzke et al. (2013)
- Lee, Wagenmakers, and Matzke (2013)
- Walker, Hickok, and Fridriksson (2018)

References

Batchelder, William H, and David M Riefer. 1999. "Theoretical and Empirical Review of Multinomial Process Tree Modeling." *Psychonomic Bulletin & Review* 6 (1). Springer: 57–86.

Lee, Michael D, Eric-Jan Wagenmakers, and Dora Matzke. 2013. "Multinomial Processing Trees." In *Bayesian Cognitive Modeling: A Practical Course*, by Michael D Lee and Eric-Jan Wagenmakers. Cambridge University Press.

Matzke, Dora, Conor V Dolan, William H Batchelder, and Eric-Jan Wagenmakers. 2013. "Bayesian Estimation of Multinomial Processing Tree Models with Heterogeneity in Participants and Items." *Psychometrika*. Springer, 1–31.

Walker, Grant M, Gregory Hickok, and Julius Fridriksson. 2018. "A Cognitive Psychometric Model for Assessment of Picture Naming Abilities in Aphasia." *Psychological Assessment*. American Psychological Association.