Chapter 5: Hierarchical models

Shravan Vasishth (vasishth.github.io)

July 2025

Contents

Textbook **1**

A simple linear model (EEG data): "Fixed-effects" model **2**

No-pooling model (separate linear models for each subject) **3**

Varying intercepts and varying slopes model (without correlation) **4**

Varying intercepts and varying slopes (with correlation) **11**

Varying intercept and varying slopes for subject and item ("Maximal model") **13**

Beyond the maximal model: Distributional regression **15**

A log-normal model of the Stroop effect **17**

Class exercise 1 **19**
Your tasks 20

Class exercise 2 **26**

Historical methods of data analysis and their connection to linear mixed models: t-tests and repeated measures ANOVA **29**
Some important points to note 32

Textbook

Introduction to Bayesian Data Analysis for Cognitive Science

Nicenboim, Schad, Vasishth

- Online version: https://bruno.nicenboim.me/bayescogsci/
- Source code: https://github.com/bnicenboim/bayescogsci
- Physical book: here

This lecture presupposes that you have read chapter 5 of the textbook.

A simple linear model (EEG data): "Fixed-effects" model

$$signal_n \sim Normal(\alpha + c\_cloze_n \cdot \beta, \sigma) \quad (1)$$

where $n = 1, \ldots, N$, and $signal$ is the dependent variable (average signal in the N400 spatiotemporal window in microvolts). The variable N represents the total number of data points.

We are going to use the following priors:

$$\begin{aligned} \alpha &\sim Normal(0, 10) \\ \beta &\sim Normal(0, 10) \\ \sigma &\sim Normal_+(0, 50) \end{aligned} \quad (2)$$

```
df_eeg$c_cloze<-df_eeg$cloze-mean(df_eeg$cloze)

fit_N400_cp <-
  brm(n400 ~ c_cloze,
      prior = c(prior(normal(0, 10), class = Intercept),
                prior(normal(0, 10), class = b, coef = c_cloze),
                prior(normal(0, 50), class = sigma)),
      data = df_eeg)
```

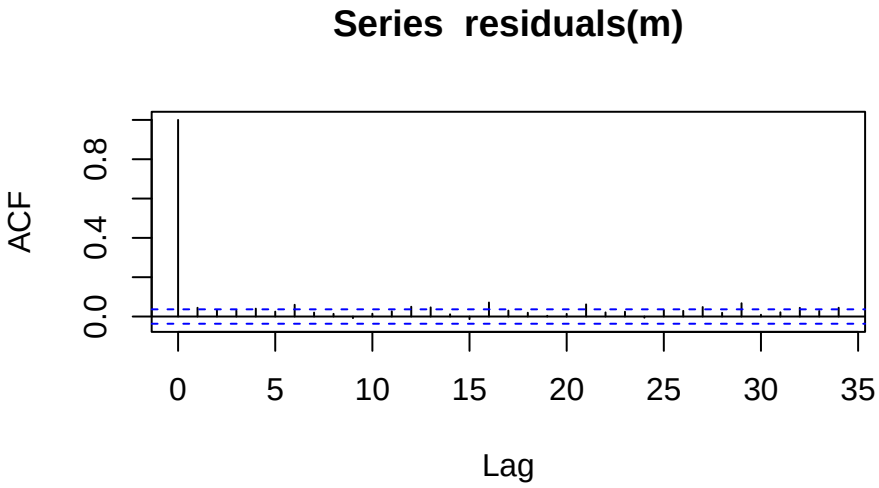
Check the summary:

```
short_summary(fit_N400_cp)
```

```
## ...
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     3.66      0.22     3.24     4.09 1.00     4119     2833
## c_cloze       2.25      0.55     1.18     3.33 1.00     4497     3207
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma    11.82      0.16    11.51    12.13 1.00     4475     3162
##
## ...
```

The correlation between the residuals (not terrible):

```
m<-lm(n400 ~ c_cloze,df_eeg)
acf(residuals(m))
```



In repeated measurements data, if you fit a fixed effects model, the residuals can be heavily correlated because one has multiple measurements from each subject and each item.

No-pooling model (separate linear models for each subject)

$$signal_n \sim Normal(\alpha_{subj[n]} + c\_cloze_n \cdot \beta_{subj[n]}, \sigma) \quad (3)$$

$$\begin{aligned} \alpha_i &\sim Normal(0, 10) \\ \beta_i &\sim Normal(0, 10) \\ \sigma &\sim Normal_+(0, 50) \end{aligned} \quad (4)$$

```
fit_N400_np <- brm(n400 ~ 0 + factor(subj) + c_cloze:factor(subj),
                   prior = c(prior(normal(0, 10), class = b),
                             prior(normal(0, 50), class = sigma)),
                   data = df_eeg)
```

```
# parameter name of beta by subject:
ind_effects_np <- paste0("b_factorsubj",
                         unique(df_eeg$subj), ":c_cloze")
beta_across_subj <- as.data.frame(fit_N400_np) %>%
  #removes the meta data from the object
  select(all_of(ind_effects_np)) %>%
  rowMeans()

# Calculate the average of these estimates
(grand_av_beta <- tibble(mean = mean(beta_across_subj),
                         lq = quantile(beta_across_subj, c(0.025)),
```

```
                          hq = quantile(beta_across_subj, c(0.975))))
```

```
## # A tibble: 1 x 3
##    mean    lq    hq
##   <dbl> <dbl> <dbl>
## 1  2.17  1.20  3.20
```

```
# make a table of beta's by subject
beta_by_subj <- posterior_summary(fit_N400_np,
                                  variable = ind_effects_np) %>%
  as.data.frame() %>%
  mutate(subject = 1:n()) %>%
  ## reorder plot by magnitude of mean:
  arrange(Estimate) %>%
  mutate(subject = factor(subject, levels = subject))
```

```
ggplot(beta_by_subj,
       aes(x = Estimate, xmin = Q2.5, xmax = Q97.5, y = subject)) +
  geom_point() +
  geom_errorbarh() +
  geom_vline(xintercept = grand_av_beta$mean) +
  geom_vline(xintercept = grand_av_beta$lq, linetype = "dashed") +
  geom_vline(xintercept = grand_av_beta$hq, linetype = "dashed") +
  xlab("By-subject effect of cloze probability in microvolts")
```

Varying intercepts and varying slopes model (without correlation)

$$signal_n \sim Normal(\alpha + u_{subj[n],1} + c\_cloze_n \cdot (\beta + u_{subj[n],2}), \sigma) \tag{5}$$

$$\begin{aligned}
\alpha &\sim Normal(0, 10) \\
\beta &\sim Normal(0, 10) \\
u_1 &\sim Normal(0, \tau_{u_1}) \\
u_2 &\sim Normal(0, \tau_{u_2}) \\
\tau_{u_1} &\sim Normal_+(0, 20) \\
\tau_{u_2} &\sim Normal_+(0, 20) \\
\sigma &\sim Normal_+(0, 50)
\end{aligned} \tag{6}$$

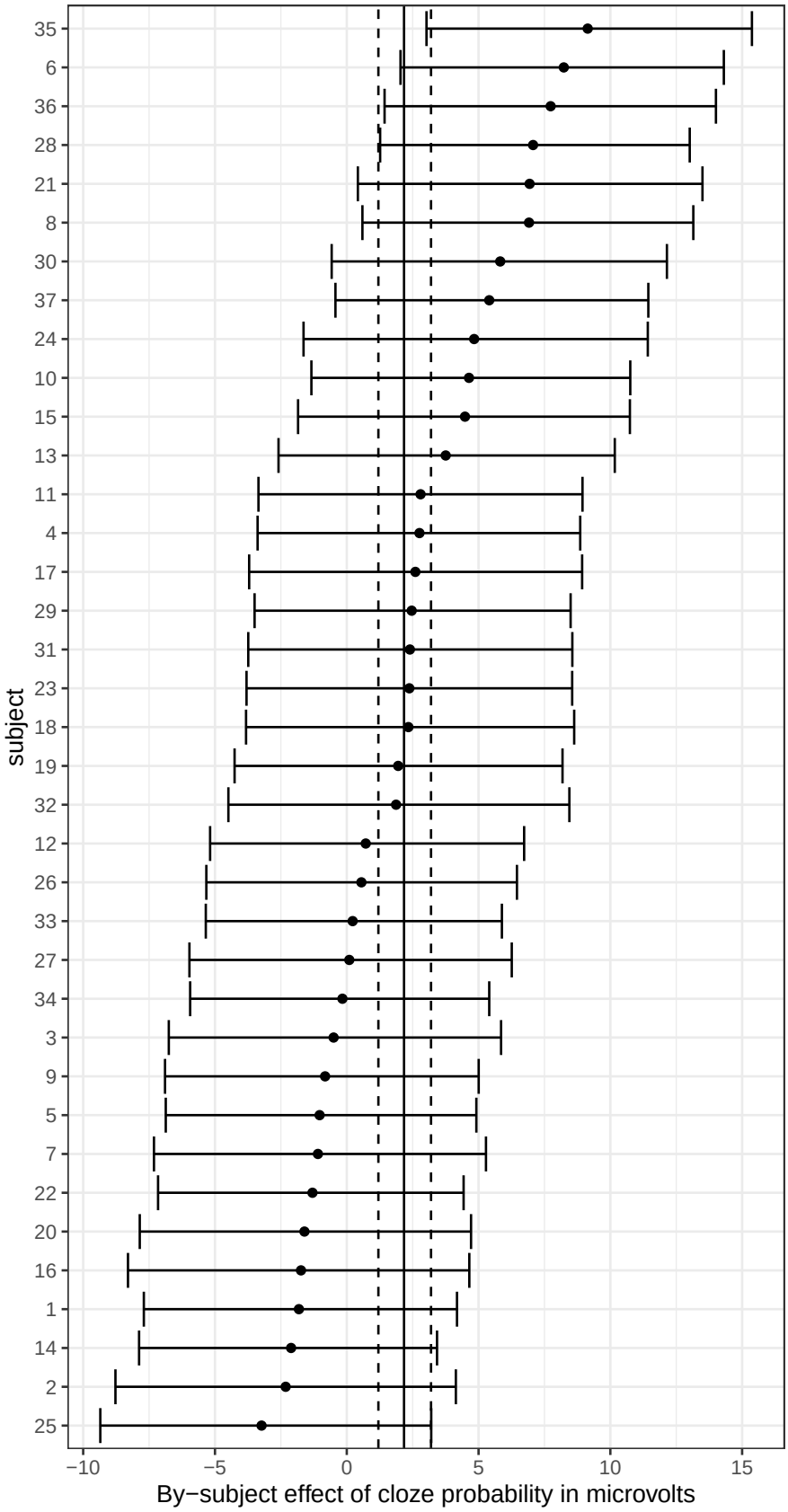


Figure 1: No pooling model.

```
prior_v <-
  c(prior(normal(0, 10), class = Intercept),
    prior(normal(0, 10), class = b, coef = c_cloze),
    prior(normal(0, 50), class = sigma),
    prior(normal(0, 20), class = sd, coef = Intercept, group = subj),
    prior(normal(0, 20), class = sd, coef = c_cloze, group = subj))
fit_N400_v <- brm(n400 ~ c_cloze + (c_cloze || subj),
                  prior = prior_v,
                  data = df_eeg)
```

When we print a `brms` fit, we first see the summaries of the posteriors of the standard deviation of the by-group intercept and slopes, τ_{u_1} and τ_{u_2} as `sd(Intercept)` and `sd(c_cloze)`, and then, as with previous models, the population-level effects, α and β as `Intercept` and `c_cloze`, and the scale of the likelihood, σ, as `sigma`. The full summary can be printed out by typing:

```
short_summary(fit_N400_v)
```

```
## ...
## Group-Level Effects:
## ~subj (Number of levels: 37)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     2.17      0.37     1.53     2.95 1.00     1351     2319
## sd(c_cloze)       1.74      0.91     0.18     3.60 1.01     1099     1446
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     3.65      0.43     2.81     4.47 1.00     1530     2146
## c_cloze       2.34      0.62     1.14     3.58 1.00     3113     2623
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma    11.62      0.16    11.32    11.94 1.00     5645     2996
##
## ...
```

```
mcmc_dens(fit_N400_v, pars = variables(fit_N400_v)[1:5])
```

```
# make a table of u_2s
ind_effects_v <- paste0("r_subj[", unique(df_eeg$subj),
                        ",c_cloze]")
u_2_v <- posterior_summary(fit_N400_v, variable = ind_effects_v) %>%
  as_tibble() %>%
  mutate(subj = 1:n()) %>%
  ## reorder plot by magnitude of mean:
  arrange(Estimate) %>%
  mutate(subj = factor(subj, levels = subj))
# We plot:
```

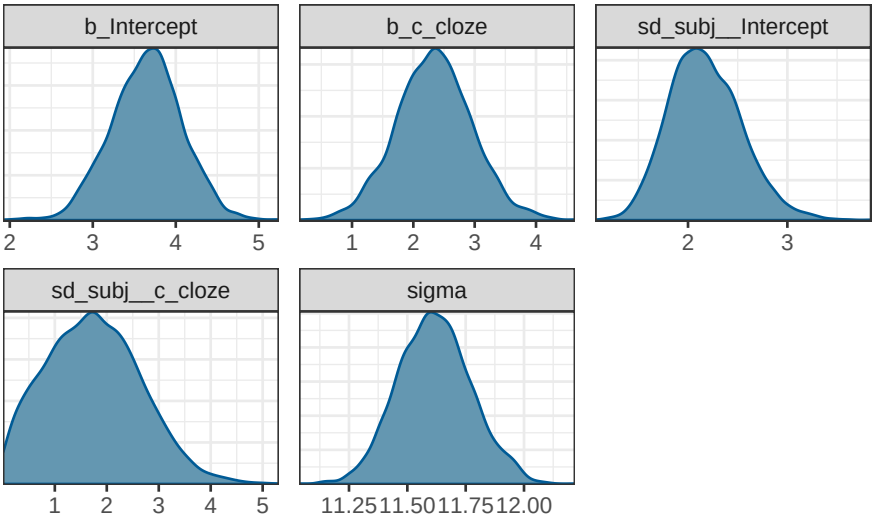


Figure 2: The posteriors of the parameters.

```
ggplot(u_2_v,
       aes(x = Estimate, xmin = Q2.5, xmax = Q97.5, y = subj)) +
  geom_point() +
  geom_errorbarh() +
  xlab("By-subject adjustment to the slope in microvolts")
```

There is an important difference between the no-pooling model and the varying intercepts and slopes model we just fit. The no-pooling model fits each individual subject's intercept and slope independently for each subject. By contrast, the varying intercepts and slopes model takes *all* the subjects' data into account in order to compute the fixed effects α and β; and the model "shrinks" the by-subject intercept and slope adjustments towards the fixed effects estimates. In Figure 4 below, we can see the shrinkage of the estimates in the varying intercepts model by comparing them with the estimates of the no pooling model (M_{np}).

```
# Extract parameter estimates from the no pooling model:
par_np <- posterior_summary(fit_N400_np, variable = ind_effects_np) %>%
  as_tibble() %>%
  mutate(model = "No pooling",
         subj = unique(df_eeg$subj))
# For the hierarchical model, the code is more complicated
# because we want the effect (beta) + adjustment.
# Extract the overall group level effect:
```

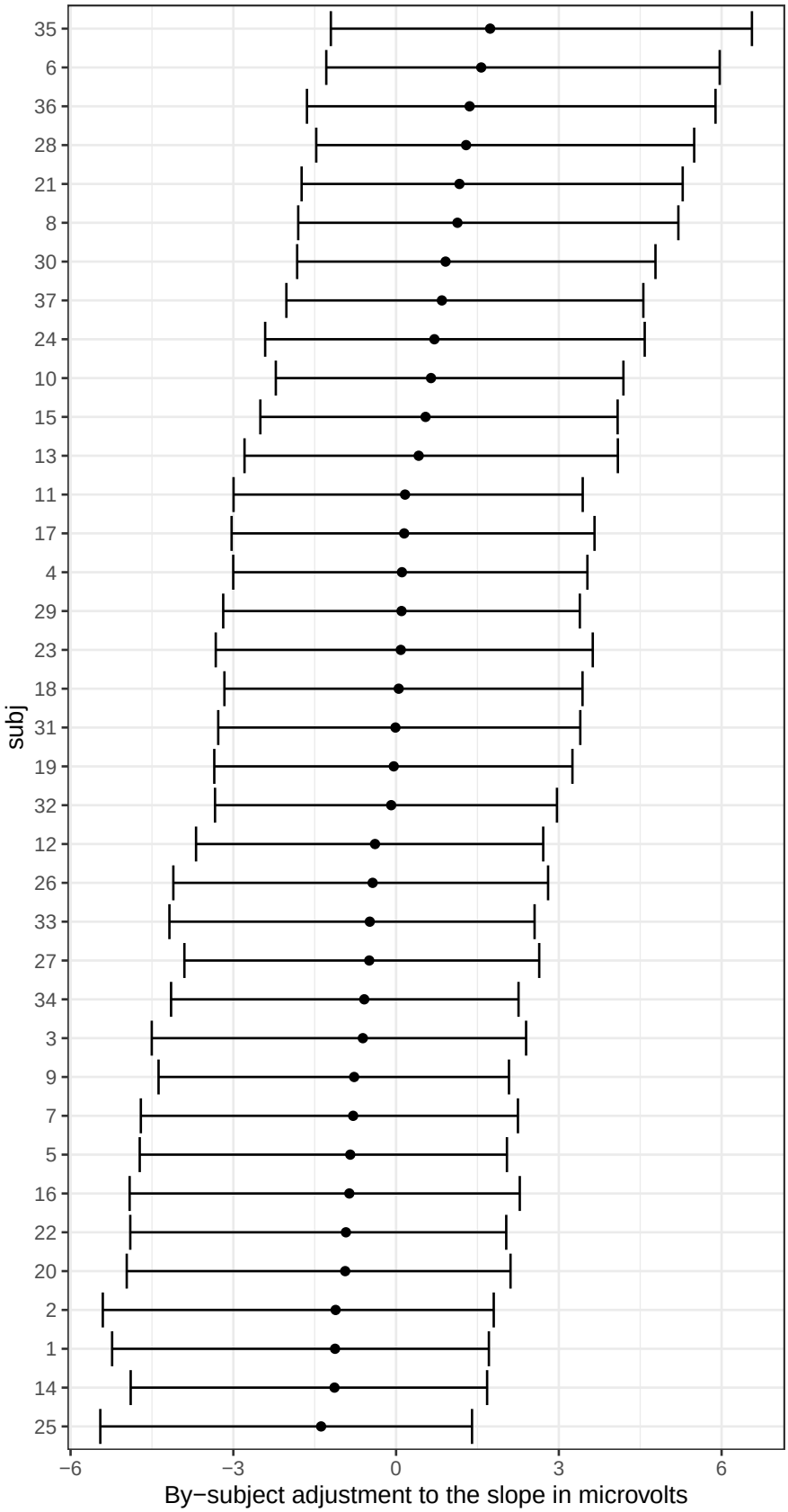


Figure 3: Partial pooling.

```r
beta <- c(as_draws_df(fit_N400_v)$b_c_cloze)
# Extract the individual adjustments:
ind_effects_v <- paste0("r_subj[", unique(df_eeg$subj), ",c_cloze]")
adjustment <- as_draws_matrix(fit_N400_v, variable = ind_effects_v)
# Get the by subject effects in a data frame where each adjustment
# is in each column.
# Remove all the draws meta data by using as.data.frame
by_subj_effect <- as.data.frame(beta + adjustment)
# Summarize them by getting a table with the mean and the
# quantiles for each column and then binding them.
par_h <- lapply(by_subj_effect, function(x) {
  tibble(Estimate = mean(x),
         Q2.5 = quantile(x, .025),
         Q97.5 = quantile(x, .975))}) %>%
  bind_rows() %>%
  # Add a column to identify that the model,
  # and one with the subject labels:
  mutate(model = "Hierarchical",
         subj = unique(df_eeg$subj))
# The mean and 95% CI of both models in one data frame:
by_subj_df <- bind_rows(par_h, par_np) %>%
  arrange(Estimate) %>%
  mutate(subj = factor(subj, levels = unique(.data$subj)))
```

```r
b_c_cloze <- posterior_summary(fit_N400_v, variable = "b_c_cloze")
ggplot(by_subj_df,
       aes(ymin = Q2.5, ymax = Q97.5, x = subj,
           y = Estimate, color = model, shape = model)) +
  geom_errorbar(position = position_dodge(1)) +
  geom_point(position = position_dodge(1)) +
  # We'll also add the mean and 95% CrI of the overall difference
  # to the plot:
  geom_hline(yintercept = b_c_cloze[, "Estimate"]) +
  geom_hline(yintercept = b_c_cloze[, "Q2.5"],
             linetype = "dotted", linewidth = .5) +
  geom_hline(yintercept = b_c_cloze[, "Q97.5"],
             linetype = "dotted", linewidth = .5) +
  xlab("N400 effect of predictability") +
  coord_flip()
```

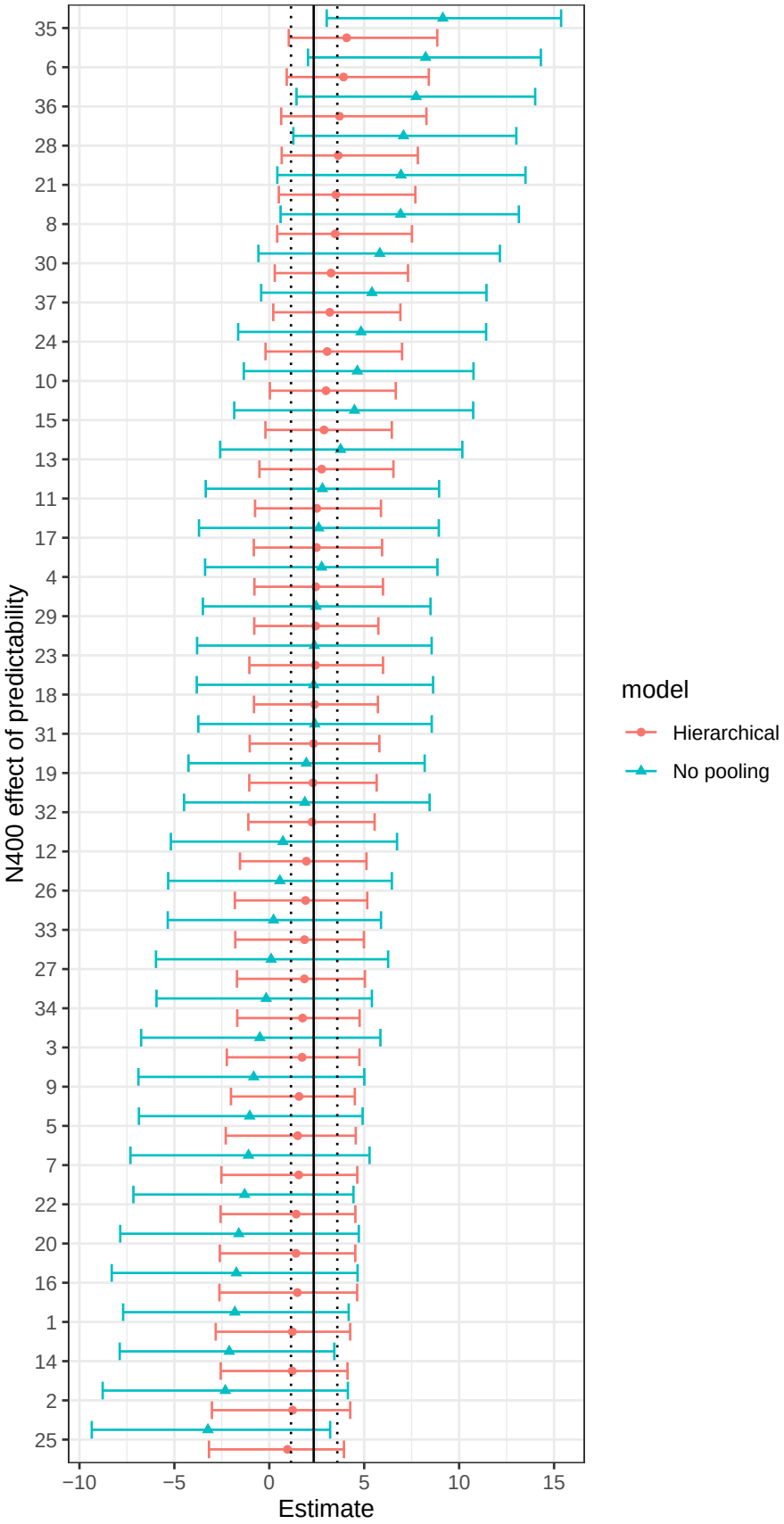


Figure 4: This plot compares the estimates of the effect of cloze probability for each subject between (i) the no pooling and (ii) the varying intercepts and varying slopes, hierarchical, model.

Varying intercepts and varying slopes (with correlation)

$$signal_n \sim Normal(\alpha + u_{subj[n],1} + c\_cloze_n \cdot (\beta + u_{subj[n],2}), \sigma) \tag{7}$$

The correlation is indicated in the priors on the adjustments for intercept u_1 and slopes u_2.

- Priors:

$$\begin{aligned} \alpha &\sim Normal(0, 10) \\ \beta &\sim Normal(0, 10) \\ \sigma &\sim Normal_+(0, 50) \end{aligned} \tag{8}$$

$$\begin{pmatrix} u_{i,1} \\ u_{i,2} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \boldsymbol{\Sigma_u} \right)$$

$$\boldsymbol{\Sigma_u} = \begin{pmatrix} \tau_{u_1}^2 & \rho_u \tau_{u_1} \tau_{u_2} \\ \rho_u \tau_{u_1} \tau_{u_2} & \tau_{u_2}^2 \end{pmatrix} \tag{9}$$

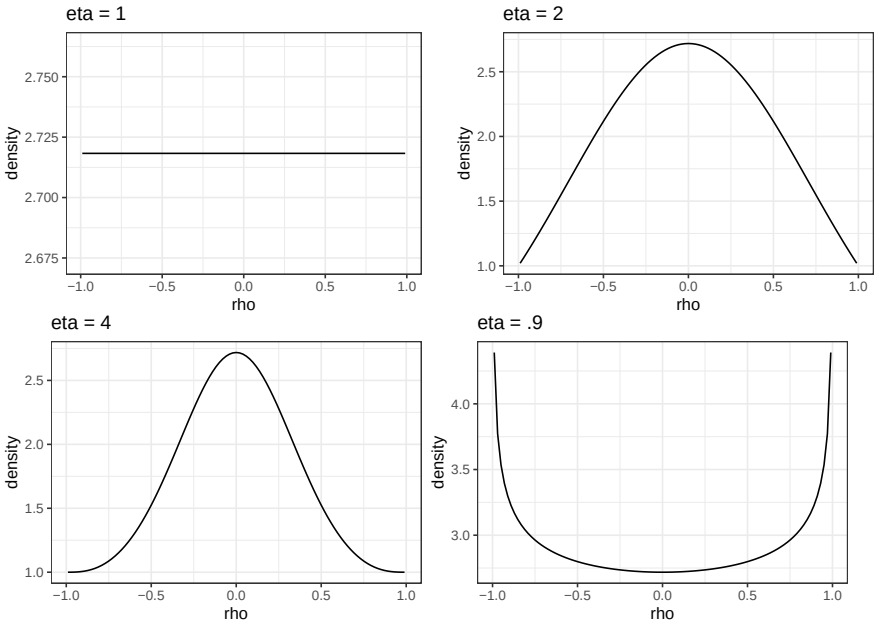


Figure 5: Visualization of the LKJ correlation distribution prior with four different values of the eta parameter.

$$\begin{aligned} \tau_{u_1} &\sim Normal_+(0, 20) \\ \tau_{u_2} &\sim Normal_+(0, 20) \\ \rho_u &\sim LKJcorr(2) \end{aligned} \tag{10}$$

```
prior_h <- c(prior(normal(0, 10), class = Intercept),
             prior(normal(0, 10), class = b, coef = c_cloze),
             prior(normal(0, 50), class = sigma),
             prior(normal(0, 20),
```

```
                    class = sd, coef = Intercept,
                    group = subj
                    ),
                prior(normal(0, 20),
                    class = sd, coef = c_cloze,
                    group = subj),
                prior(lkj(2), class = cor, group = subj))
fit_N400_h <- brm(n400 ~ c_cloze + (c_cloze | subj),
                  prior = prior_h,
                  data = df_eeg)
```

```
plot(fit_N400_h, nvariables = 6)
```

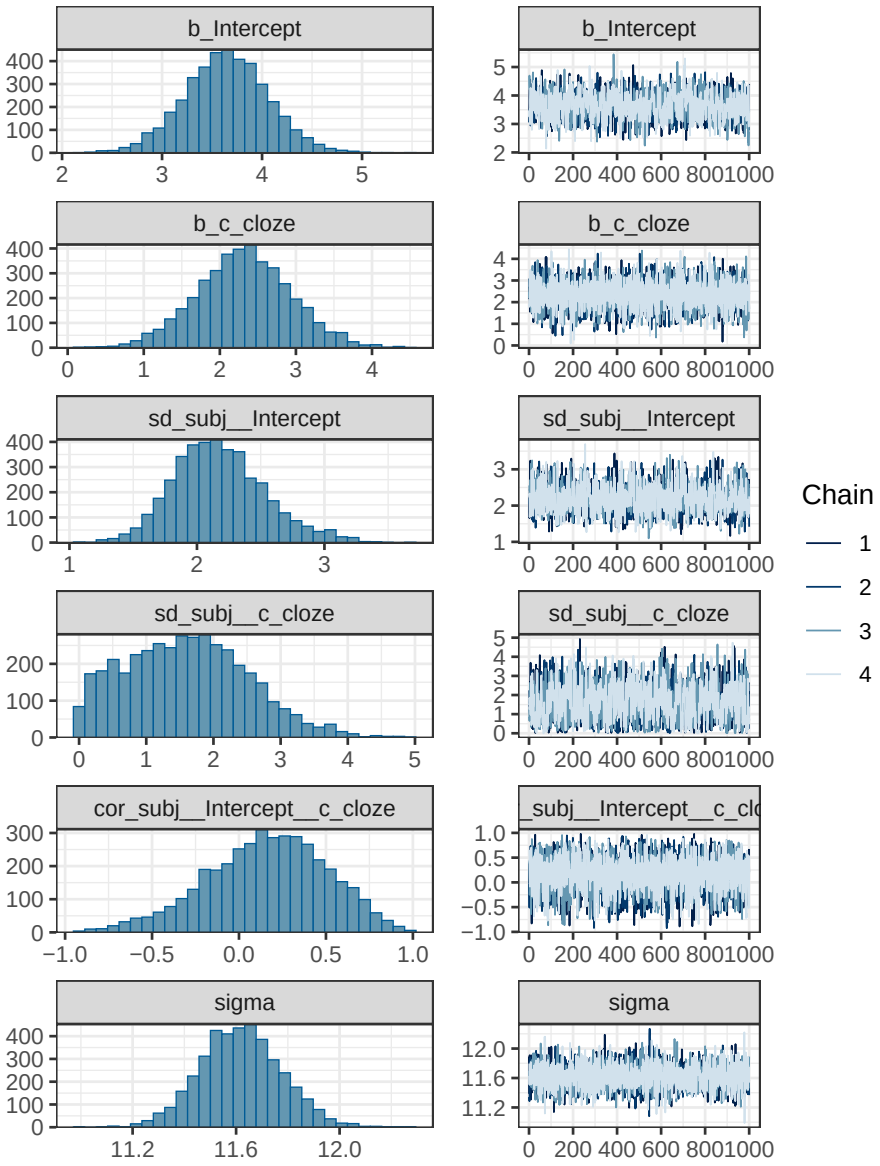


Figure 6: Posteriors.

Varying intercept and varying slopes for subject and item ("Maximal model")

The term "maximal model" became mainstream in psycholinguistics and psychology after Barr et al 2013 came out.

$$signal_n \sim Normal(\alpha + u_{subj[n],1} + w_{item[n],1} + c\_cloze_n \cdot (\beta + u_{subj[n],2} + w_{item[n],2}), \sigma) \tag{11}$$

- Priors:

$$\begin{aligned} \alpha &\sim Normal(0, 10) \\ \beta &\sim Normal(0, 10) \\ \sigma &\sim Normal_+(0, 50) \\ \begin{pmatrix} u_{i,1} \\ u_{i,2} \end{pmatrix} &\sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \boldsymbol{\Sigma_u} \right) \\ \begin{pmatrix} w_{j,1} \\ w_{j,2} \end{pmatrix} &\sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \boldsymbol{\Sigma_w} \right) \end{aligned} \tag{12}$$

We have added the index j, which represents each item, as we did with subjects; $item[n]$ indicates the item that corresponds to the observation in the n-th row of the data frame.

We have hyperparameters and hyperpriors as before:

$$\begin{aligned} \boldsymbol{\Sigma_u} &= \begin{pmatrix} \tau_{u_1}^2 & \rho_u \tau_{u_1} \tau_{u_2} \\ \rho_u \tau_{u_1} \tau_{u_2} & \tau_{u_2}^2 \end{pmatrix} \\ \boldsymbol{\Sigma_w} &= \begin{pmatrix} \tau_{w_1}^2 & \rho_w \tau_{w_1} \tau_{w_2} \\ \rho_w \tau_{w_1} \tau_{w_2} & \tau_{w_2}^2 \end{pmatrix} \end{aligned} \tag{13}$$

$$\begin{aligned} \tau_{u_1} &\sim Normal_+(0, 20) \\ \tau_{u_2} &\sim Normal_+(0, 20) \\ \rho_u &\sim LKJcorr(2) \\ \tau_{w_1} &\sim Normal_+(0, 20) \\ \tau_{w_2} &\sim Normal_+(0, 20) \\ \rho_w &\sim LKJcorr(2) \end{aligned} \tag{14}$$

```
prior_sih_full <-
  c(prior(normal(0, 10), class = Intercept),
    prior(normal(0, 10), class = b, coef = c_cloze),
```

```
    prior(normal(0, 50), class = sigma),
    prior(normal(0, 20),
          class = sd, coef = Intercept,
          group = subj),
    prior(normal(0, 20),
          class = sd, coef = c_cloze,
          group = subj),
    prior(lkj(2), class = cor, group = subj),
    prior(normal(0, 20),
          class = sd, coef = Intercept,
          group = item),
    prior(normal(0, 20),
          class = sd, coef = c_cloze,
          group = item),
    prior(lkj(2), class = cor, group = item))
```

```
fit_N400_sih <- brm(n400 ~ c_cloze + (c_cloze | subj) +
                      (c_cloze | item),
                    prior = prior_sih_full,
                    data = df_eeg)
```

We can also simplify the call to `brms`, when we assign the same priors to the by-subject and by-item parameters:

```
prior_sih <-
  c(prior(normal(0, 10), class = Intercept),
    prior(normal(0, 10), class = b),
    prior(normal(0, 50), class = sigma),
    prior(normal(0, 20), class = sd),
    prior(lkj(2), class = cor))
```

```
fit_N400_sih <- brm(n400 ~ c_cloze + (c_cloze | subj) +
                      (c_cloze | item),
                    prior = prior_sih,
                    data = df_eeg)
```

```
short_summary(fit_N400_sih)
```

```
## ...
## Group-Level Effects:
## ~item (Number of levels: 80)
##                       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)             1.51      0.34     0.82     2.18 1.00     1476
## sd(c_cloze)               2.31      0.98     0.38     4.13 1.00     1085
## cor(Intercept,c_cloze)   -0.42      0.31    -0.90     0.28 1.00     1829
```

```
##                         Tail_ESS
## sd(Intercept)               1686
## sd(c_cloze)                 1207
## cor(Intercept,c_cloze)      2157
##
## ~subj (Number of levels: 37)
##                         Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)               2.20      0.36     1.59     2.98 1.00     1380
## sd(c_cloze)                 1.54      0.90     0.11     3.38 1.01      932
## cor(Intercept,c_cloze)      0.13      0.35    -0.59     0.78 1.00     3082
##                         Tail_ESS
## sd(Intercept)               2221
## sd(c_cloze)                 1440
## cor(Intercept,c_cloze)      2397
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     3.65      0.47     2.72     4.61 1.00     1169     2067
## c_cloze       2.32      0.69     0.91     3.72 1.00     2685     2658
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma    11.48      0.16    11.18    11.80 1.00     4589     2893
##
## ...
```

```
plot(fit_N400_sih, nvariables = 9)
```

Beyond the maximal model: Distributional regression

$$
\begin{aligned}
signal_n \sim Normal(&\alpha + u_{subj[n],1} + w_{item[n],1} + \\
&c\_cloze_n \cdot (\beta + u_{subj[n],2} + w_{item[n],2}), \sigma_n) \\
\sigma_n = \exp(&\sigma_\alpha + \sigma_{u_{subj[n]}})
\end{aligned}
\tag{15}
$$

$$
\begin{aligned}
\sigma_\alpha &\sim Normal(0, log(50)) \\
\sigma_u &\sim Normal(0, \tau_{\sigma_u}) \\
\tau_{\sigma_u} &\sim Normal_+(0, 5)
\end{aligned}
\tag{16}
$$

```
prior_s <-
  c(prior(normal(0, 10), class = Intercept),
    prior(normal(0, 10), class = b),
    prior(normal(0, 20), class = sd),
    prior(lkj(2), class = cor),
    prior(normal(0, log(50)), class = Intercept, dpar = sigma),
    prior(normal(0, 5),
          class = sd, group = subj,
          dpar = sigma))
```

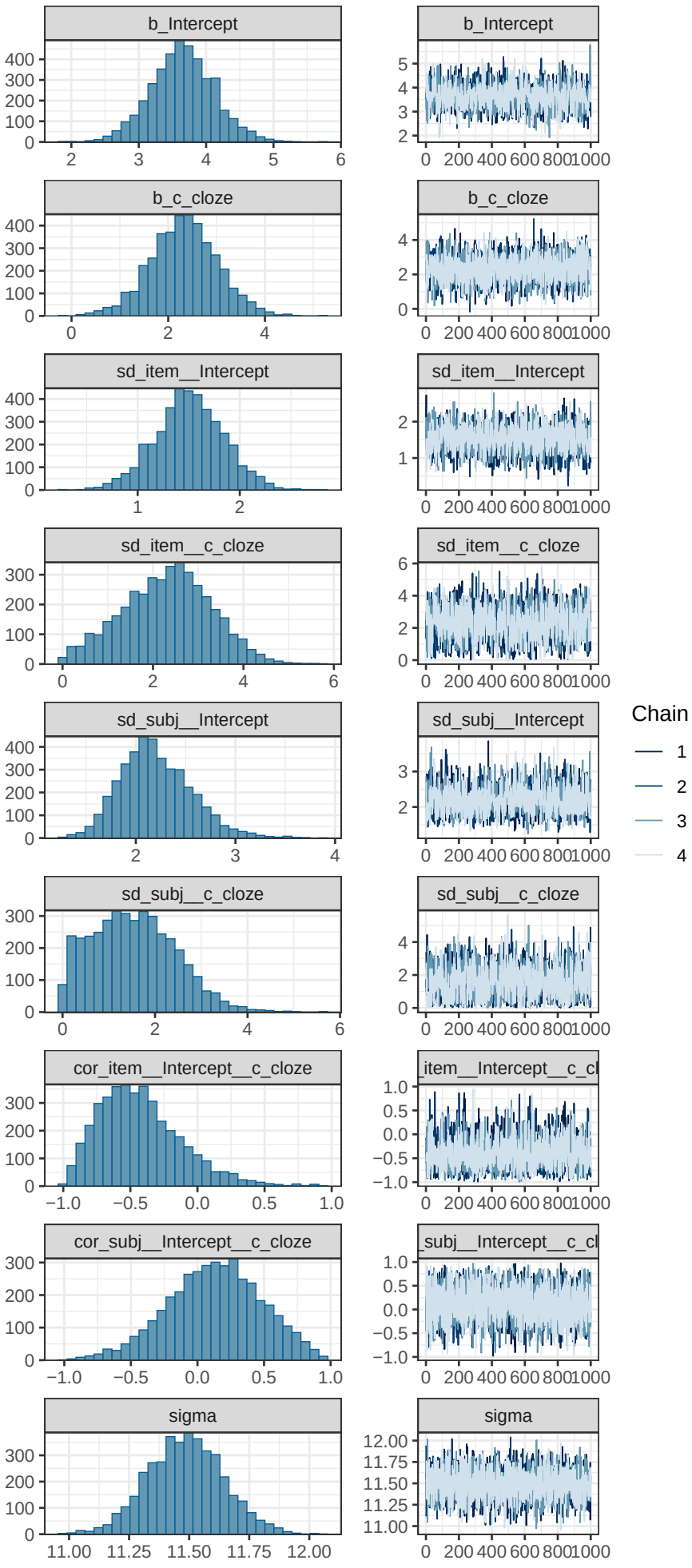


Figure 7: The posterior distributions of the parameters in the model with varying intercepts and varying slopes for subjects and items.

```
fit_N400_s <- brm(brmsformula(
  n400 ~ c_cloze + (c_cloze | subj) + (c_cloze | item),
  sigma ~ 1 + (1 | subj)),
  prior = prior_s, data = df_eeg)
```

Inspect the output below; notice that our estimate for the effect of cloze remains very similar to that of the model `fit_N400_sih`.

Compare the two models' estimates:

```
posterior_summary(fit_N400_sih, variable = "b_c_cloze")
```

```
##           Estimate Est.Error      Q2.5    Q97.5
## b_c_cloze 2.323392 0.6943517 0.9098678 3.715103
```

```
posterior_summary(fit_N400_s, variable = "b_c_cloze")
```

```
##           Estimate Est.Error      Q2.5    Q97.5
## b_c_cloze 2.287663 0.6682787 0.9463829 3.635023
```

A log-normal model of the Stroop effect

$$rt_n \sim LogNormal(\alpha + u_{subj[n],1} + c\_cond_n \cdot (\beta + u_{subj[n],2}), \sigma) \tag{17}$$

$$\begin{aligned} \mu_{incongruent} &= \alpha + 1 \cdot \beta \\ \mu_{congruent} &= \alpha + -1 \cdot \beta \end{aligned} \tag{18}$$

$$\begin{aligned} \alpha &\sim Normal(6, 1.5) \\ \beta &\sim Normal(0, 0.01) \\ \sigma &\sim Normal_+(0, 1) \end{aligned} \tag{19}$$

$$\begin{pmatrix} u_{i,1} \\ u_{i,2} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \boldsymbol{\Sigma_u}\right) \tag{20}$$

$$\boldsymbol{\Sigma_u} = \begin{pmatrix} \tau_{u_1}^2 & \rho_u \tau_{u_1} \tau_{u_2} \\ \rho_u \tau_{u_1} \tau_{u_2} & \tau_{u_2}^2 \end{pmatrix} \tag{21}$$

$$\begin{aligned} \tau_{u_1} &\sim Normal_+(0, 1) \\ \tau_{u_2} &\sim Normal_+(0, 1) \\ \rho_u &\sim LKJcorr(2) \end{aligned} \tag{22}$$

We restrict ourselves to the correct trials only, and add a `c_cond` predictor, sum-coded as described earlier.

```
data("df_stroop")
(df_stroop <- df_stroop %>%
  mutate(c_cond = if_else(condition == "Incongruent", 1, -1)))
```

```
## # A tibble: 3,058 x 5
##     subj trial condition      RT c_cond
##    <dbl> <int> <chr>       <int>  <dbl>
##  1     1     0 Congruent    1484     -1
##  2     1     1 Incongruent  1316      1
##  3     1     2 Incongruent   628      1
##  4     1     3 Congruent     511     -1
##  5     1     4 Congruent     509     -1
##  6     1     5 Incongruent   903      1
##  7     1     6 Incongruent   759      1
##  8     1     7 Incongruent  1133      1
##  9     1     8 Incongruent   678      1
## 10     1     9 Incongruent   785      1
## # i 3,048 more rows
```

Fit the model.

```
fit_stroop <- brm(RT ~ c_cond + (c_cond | subj),
                  family = lognormal(),
                  prior =
                    c(prior(normal(6, 1.5), class = Intercept),
                      prior(normal(0, .01), class = b),
                      prior(normal(0, 1), class = sigma),
                      prior(normal(0, 1), class = sd),
                      prior(lkj(2), class = cor)),
                  data = df_stroop)
```

We will focus on β (but you can verify that there is nothing surprising in the other parameters in the model `fit_stroop`).

```
posterior_summary(fit_stroop, variable = "b_c_cond")
```

```
##             Estimate   Est.Error       Q2.5      Q97.5
## b_c_cond 0.02711678 0.005411854 0.01589884 0.03730112
```

Class exercise 1

This exercise tests your ability to map the `lme4` syntax for linear mixed models to the underlying mathematical/statistical model. This skill is important because in later chapters we will go from the mathematical statement of the model to Stan syntax.

The Dillon et al 2013 data set looks at interference effects in reflexives in English. We look at a simplified version of these data.

```
library(bcogsci)
data("df_dillonE1")
head(df_dillonE1)
```

```
##         subj       item   rt int     expt
## 49 dillonE11 dillonE119 2918 low dillonE1
## 56 dillonE11 dillonE119 1338 low dillonE1
## 63 dillonE11 dillonE119  424 low dillonE1
## 70 dillonE11 dillonE119  186 low dillonE1
## 77 dillonE11 dillonE119  195 low dillonE1
## 84 dillonE11 dillonE119 1218 low dillonE1
```

I have removed six other conditions, and focus only on two conditions (so this design is not completely correct; we will return to the full design after the contrast coding lectures).

The two conditions are as follows (both are ungrammatical sentences):

- High interference: The amateur bodybuilder who worked with the *personal trainers* amazingly injured *themselves* on the lightest weights.
- Low interference: The amateur bodybuilder who worked with the *personal trainer* amazingly injured *themselves* on the lightest weights.

Theory predicts a difference between high and low interference conditions in reading time, at the word themselves. The reason is that there is an illusion of grammaticality in the high interference condition due to local agreement between trainers and themselves. For details, see the original paper.

If we compute grand averages, we see that high interference conditions are indeed read faster than low interference conditions:

```
round(with(df_dillonE1,tapply(rt,int,mean)))
```

```
## high  low
##  801  856
```

Your tasks

- First, set up an appropriate contrast coding for this design: +1/2 for high interference and -1/2 for low interference:

```
df_dillonE1$c_int<-ifelse(df_dillonE1$int=="low",-1/2,1/2)
```

Fit the following models in brms. You will have to decide on priors. **For each model, write down the underlying statistical model in mathematical notation**.

- A no pooling model analogous to this frequentist one:

```
library(lme4)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'lme4'
```

```
## The following object is masked from 'package:brms':
##
##     ngrps
```

```
m1<-lmList(log(rt)~c_int|subj,df_dillonE1)
## separate estimates
m1
```

```
## Call: lmList(formula = log(rt) ~ c_int | subj, data = df_dillonE1)
## Coefficients:
##             (Intercept)        c_int
## dillonE11     6.964055  0.162119114
## dillonE111    6.697648  0.099414490
## dillonE112    6.601500  0.188842100
## dillonE114    6.409764 -0.374700251
```

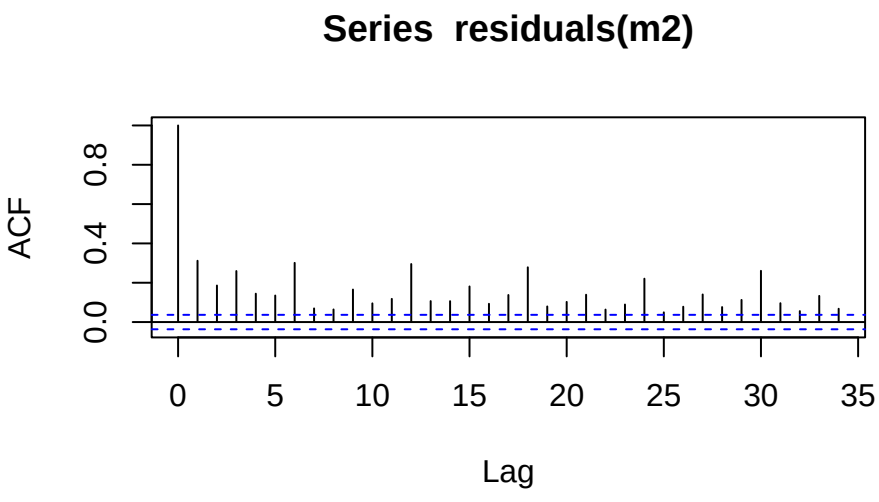
```
## dillonE116    6.296188 -0.402684377
## dillonE117    6.003641  0.108183364
## dillonE12     6.390891 -0.006007199
## dillonE123    6.517746  0.024104734
## dillonE124    6.680429 -0.049381637
## dillonE126    6.601036  0.169541977
## dillonE127    6.310201  0.093376109
## dillonE128    6.209580 -0.356135796
## dillonE129    6.590704 -0.046769730
## dillonE13     6.611758 -0.066017984
## dillonE130    6.166103 -0.139666584
## dillonE131    6.317039 -0.423852858
## dillonE132    6.814438 -0.124328800
## dillonE133    6.473360 -0.342476166
## dillonE134    6.627406  0.154700915
## dillonE135    6.539288  0.475863979
## dillonE136    6.340895 -0.241241058
## dillonE137    6.078482 -0.111717409
## dillonE138    6.907188 -0.152755829
## dillonE139    6.811871  0.056525504
## dillonE14     6.723666 -0.217727776
## dillonE140    6.908356 -0.047033060
## dillonE141    6.302499 -0.157205186
## dillonE142    6.300375 -0.078902006
## dillonE143    6.477288 -0.034400977
## dillonE144    6.543147 -0.180240290
## dillonE145    6.985113 -0.515629374
## dillonE146    6.545404 -0.103850529
## dillonE147    6.579029  0.163569924
## dillonE148    6.954835 -0.196034490
## dillonE149    6.577471 -0.043719083
## dillonE15     6.679635  0.165247144
## dillonE150    6.382706 -0.161423042
## dillonE16     6.594270 -0.043360075
## dillonE17     5.623231  0.171543369
## dillonE19     6.330327  0.137291840
##
## Degrees of freedom: 2855 total; 2775 residual
## Residual standard error: 0.5835455
```

- A complete pooling model

```r
m2<-lm(log(rt)~c_int,df_dillonE1)
summary(m2)
```

```
##
## Call:
## lm(formula = log(rt) ~ c_int, data = df_dillonE1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.96031 -0.42850  0.02082  0.45014  2.68898
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.51257    0.01210 538.130   <2e-16 ***
## c_int       -0.06093    0.02420  -2.517   0.0119 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6466 on 2853 degrees of freedom
## Multiple R-squared:  0.002216,   Adjusted R-squared:  0.001866
## F-statistic: 6.337 on 1 and 2853 DF,  p-value: 0.01188
```

```r
## Here is one reason why this model is wrong:
acf(residuals(m2))
```



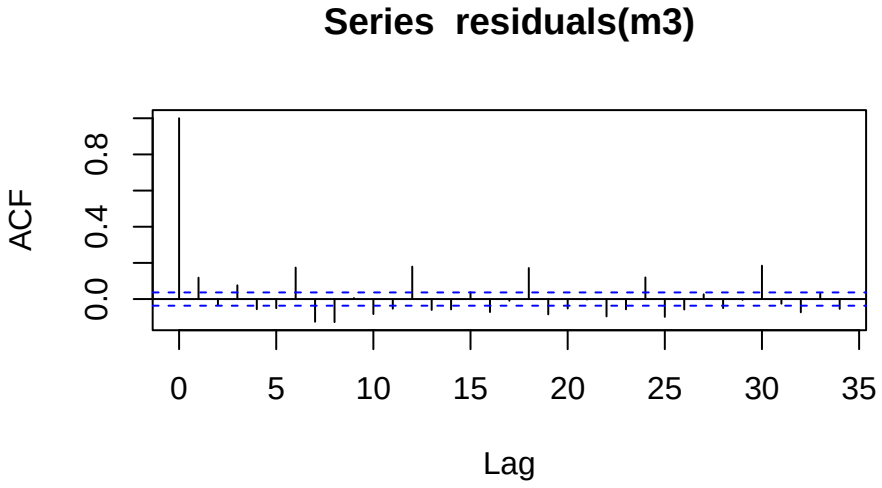
- A linear mixed model with varying intercepts for subject and for item

```r
m3<-lmer(log(rt)~c_int+(1|subj)+(1|item),df_dillonE1)
summary(m3)
```

```
## Linear mixed model fit by REML ['lmerMod']
```

```
## Formula: log(rt) ~ c_int + (1 | subj) + (1 | item)
##    Data: df_dillonE1
##
## REML criterion at convergence: 5127.6
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.6855 -0.6669  0.0257  0.6750  3.7855
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  item     (Intercept) 0.01729  0.1315
##  subj     (Intercept) 0.07378  0.2716
##  Residual             0.33070  0.5751
## Number of obs: 2855, groups:  item, 48; subj, 40
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  6.51608    0.04819 135.221
## c_int       -0.05783    0.02183  -2.649
##
## Correlation of Fixed Effects:
##       (Intr)
## c_int -0.001
```

```
## look at how the acf changes compared to m2:
acf(residuals(m3))
```



- A linear mixed model with varying intercepts and varying slopes for subject and for item, with no correlation

```
m4<-lmer(log(rt)~c_int+(1+c_int||subj)+(1+c_int||item),df_dillonE1)
summary(m4)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log(rt) ~ c_int + ((1 | subj) + (0 + c_int | subj)) + ((1 | item)
##     (0 + c_int | item))
##    Data: df_dillonE1
##
## REML criterion at convergence: 5116.5
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.7619 -0.6763  0.0356  0.6722  3.7978
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  item     c_int       0.01247  0.1117
##  item.1   (Intercept) 0.01585  0.1259
##  subj     c_int       0.01399  0.1183
##  subj.1   (Intercept) 0.07285  0.2699
##  Residual             0.32523  0.5703
## Number of obs: 2855, groups:  item, 48; subj, 40
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  6.51483    0.04764 136.753
## c_int       -0.05872    0.03292  -1.783
##
## Correlation of Fixed Effects:
##       (Intr)
## c_int -0.001
```

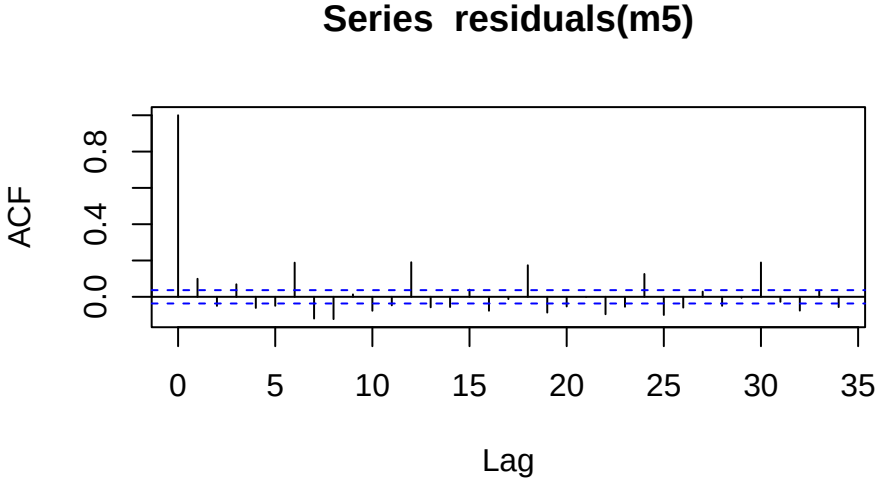
- A linear mixed model with varying intercepts and varying slopes for subject and for item, with correlation

```
m5<-lmer(log(rt)~c_int+(1+c_int|subj)+(1+c_int|item),df_dillonE1)
summary(m5)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log(rt) ~ c_int + (1 + c_int | subj) + (1 + c_int | item)
##    Data: df_dillonE1
```

```
##
## REML criterion at convergence: 5114.7
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.8132 -0.6732  0.0359  0.6777  3.8199
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  item     (Intercept) 0.01611  0.1269
##           c_int       0.01233  0.1110   -0.34
##  subj     (Intercept) 0.07387  0.2718
##           c_int       0.01380  0.1175   -0.20
##  Residual             0.32517  0.5702
## Number of obs: 2855, groups:  item, 48; subj, 40
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  6.51471    0.04796 135.827
## c_int       -0.05832    0.03280  -1.778
##
## Correlation of Fixed Effects:
##       (Intr)
## c_int -0.165
```

```
## look at how the acf changes compared to m3:
acf(residuals(m5))
```



Notice that the effect disappears in the above models when the varying slopes are added—why is that? Hint: Barr et al 2013.

Class exercise 2

Here is a 2 × 2 repeated measures factorial design:

```
data("df_smithE1")
head(df_smithE1)
```

```
##     Participant StimSet  RT N2Factor SemFactor
## 203         101      32 976     N2pl    SemSim
## 245         101      20 640     N2pl    SemSim
## 277         101       7 448     N2sg    SemSim
## 304         101      10 640     N2pl SemDissim
## 344         101      14 448     N2pl SemDissim
## 383         101      18 432     N2pl SemDissim
```

There are two factors: Semantic similarity (levels: similar and dissimilar), and the number feature on the second noun (singular vs plural):

a. Dissimilar, N2 singular: The canoe by the cabin likely *was* damaged in the heavy storm.
b. Dissimilar, N2 plural: The canoe by the cabins likely *was* damaged in the heavy storm.
c. Similar, N2 singular: The canoe by the kayak likely *was* damaged in the heavy storm.
d. Similar, N2 plural: The canoe by the kayaks likely *was* damaged in the heavy stor

The predictions (from a sentence processing theory) are:

- Main effect of similarity
- Interaction between similarity and number (larger slowdown due to singular second noun in similar conditions than in dissimilar conditions).

The original paper (their experiment 1) is:

Smith, G., Franck, J., & Tabor, W. (2021). Encoding interference effects support self-organized sentence processing. Cognitive Psychology, 124, 101356.

The following model would be appropriate (I will discuss the warning in class):

```
df_smithE1$N2<-ifelse(df_smithE1$N2Factor=="N2sg",1/2,-1/2)
df_smithE1$Sem<-ifelse(df_smithE1$SemFactor=="SemSim",1/2,-1/2)
df_smithE1$N2xSem<-df_smithE1$N2*df_smithE1$Sem
m<-lmer(log(RT)~N2+Sem+N2xSem+
          (1+N2+Sem+N2xSem|Participant)+
          (1+N2+Sem+N2xSem|StimSet),
        df_smithE1)
```

```
## boundary (singular) fit: see help('isSingular')
```

```
summary(m)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log(RT) ~ N2 + Sem + N2xSem + (1 + N2 + Sem + N2xSem | Participan
##     (1 + N2 + Sem + N2xSem | StimSet)
##    Data: df_smithE1
##
## REML criterion at convergence: 2989.1
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.3578 -0.5803 -0.1202  0.4073  8.3840
##
## Random effects:
##  Groups      Name        Variance  Std.Dev. Corr
##  Participant (Intercept) 0.0563127 0.23730
##              N2          0.0025518 0.05052  -0.13
##              Sem         0.0010239 0.03200   0.15 -0.66
##              N2xSem      0.0312132 0.17667  -0.23  0.55 -0.99
##  StimSet     (Intercept) 0.0049071 0.07005
##              N2          0.0005055 0.02248   0.21
##              Sem         0.0006671 0.02583   0.22  1.00
##              N2xSem      0.0049601 0.07043  -0.99 -0.31 -0.32
##  Residual                0.1226560 0.35022
## Number of obs: 3441, groups:  Participant, 110; StimSet, 36
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  6.03753    0.02617 230.702
## N2          -0.01765    0.01348  -1.310
## Sem          0.02927    0.01312   2.230
```

```
## N2xSem      -0.02596    0.03163  -0.821
##
## Correlation of Fixed Effects:
##         (Intr) N2     Sem
## N2      -0.015
## Sem      0.069  0.046
## N2xSem -0.266  0.091 -0.167
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

The `brms` output may be more transparent. In brms (using default priors!):

```
m_brms<-brm(log(RT)~N2+Sem+N2xSem+
              (1+N2+Sem+N2xSem|Participant)+
              (1+N2+Sem+N2xSem|StimSet),
            df_smithE1,
            family=lognormal())
```

```
short_summary(m_brms)
```

```
## ...
## Group-Level Effects:
## ~Participant (Number of levels: 110)
##                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)           0.04      0.00     0.04     0.05 1.01      736
## sd(N2)                  0.01      0.00     0.00     0.01 1.00      898
## sd(Sem)                 0.00      0.00     0.00     0.01 1.00     1233
## sd(N2xSem)              0.03      0.01     0.01     0.04 1.00      777
## cor(Intercept,N2)      -0.05      0.34    -0.71     0.66 1.00     5003
## cor(Intercept,Sem)      0.05      0.38    -0.68     0.76 1.00     6349
## cor(N2,Sem)            -0.07      0.44    -0.82     0.78 1.00     2746
## cor(Intercept,N2xSem)  -0.23      0.18    -0.57     0.12 1.00     3335
## cor(N2,N2xSem)          0.17      0.41    -0.69     0.84 1.00      613
## cor(Sem,N2xSem)        -0.27      0.41    -0.89     0.67 1.00      616
##                     Tail_ESS
## sd(Intercept)           1541
## sd(N2)                  1620
## sd(Sem)                 2238
## sd(N2xSem)               483
## cor(Intercept,N2)       1963
## cor(Intercept,Sem)      2674
## cor(N2,Sem)             3239
## cor(Intercept,N2xSem)   2238
## cor(N2,N2xSem)           801
## cor(Sem,N2xSem)         1227
##
## ~StimSet (Number of levels: 36)
##                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)           0.01      0.00     0.01     0.02 1.00     1657
## sd(N2)                  0.00      0.00     0.00     0.01 1.00     1563
## sd(Sem)                 0.00      0.00     0.00     0.01 1.00     1705
## sd(N2xSem)              0.01      0.01     0.00     0.02 1.00     1875
## cor(Intercept,N2)       0.03      0.39    -0.73     0.77 1.00     6039
## cor(Intercept,Sem)      0.07      0.38    -0.71     0.76 1.00     5680
## cor(N2,Sem)             0.04      0.44    -0.79     0.82 1.00     3233
## cor(Intercept,N2xSem)  -0.53      0.30    -0.93     0.24 1.00     3681
```

```
## cor(N2,N2xSem)           -0.03      0.42    -0.79     0.76 1.00     3893
## cor(Sem,N2xSem)          -0.09      0.43    -0.82     0.75 1.00     3277
##                       Tail_ESS
## sd(Intercept)             2473
## sd(N2)                    1792
## sd(Sem)                   1933
## sd(N2xSem)                1623
## cor(Intercept,N2)         2351
## cor(Intercept,Sem)        2561
## cor(N2,Sem)               2909
## cor(Intercept,N2xSem)     2420
## cor(N2,N2xSem)            3308
## cor(Sem,N2xSem)           2808
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     1.80      0.00     1.79     1.80 1.00      554      859
## N2           -0.00      0.00    -0.01     0.00 1.00     4769     3448
## Sem           0.00      0.00     0.00     0.01 1.00     4571     2856
## N2xSem       -0.00      0.01    -0.01     0.01 1.00     4221     3071
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.06      0.00     0.05     0.06 1.00     5425     2953
##
## ...
```

The variance-covariance matrices

Write down the full model, along with the variance-covariance matrices for the Participant and StimSet random effects.

Historical methods of data analysis and their connection to linear mixed models: t-tests and repeated measures ANOVA

Function for outputting t-test results in a more human-readable and more compact manner:

```r
summary_ttest <- function(res, paired = TRUE, units = "ms") {
  obs_t <- round(res$statistic, 2)
  dfs <- round(res$parameter)
  pval <- round(res$p.value, 3)
  ci <- round(res$conf.int, 2)
  est <- round(res$estimate, 2)
  if (paired == TRUE) {
    print(paste(
      paste("t(", dfs, ")=",
        obs_t,
        sep = ""
      ),
```

```
      paste("p=", pval, sep = "")))
      print(paste("est.: ", est, " [",
                  ci[1], ",", ci[2], "] ",
                  units, sep = "")
    )
  } else {
    print(paste(
      paste("t(", dfs, ")=",
        obs_t,
        sep = ""
      ),
      paste("p=", pval, sep = "")))
      print(paste(paste("est. 1: ", est[1], sep = ""),
      paste("est. 2: ", est[2], sep = ""),
      paste("CI of diff. in means: [", ci[1], ",", ci[2], "]", sep = "")))
  }
}
```

Paired t-test are the traditional way to analyze such data. Here's how this is done, and here is the connection to linear mixed models.

```
## all four levels: needed for interaction below:
bysubj <- aggregate(log(RT) ~ Participant +
  N2Factor + SemFactor,
mean,
data = df_smithE1
)

colnames(bysubj)[4]<-"LogRT"

bysubjN2 <- aggregate(log(RT) ~ Participant +
  N2Factor,
mean,
data = df_smithE1
)

bysubjSem <- aggregate(log(RT) ~ Participant +
  SemFactor,
mean,
data = df_smithE1
```

```
)

colnames(bysubjN2)[3]<-"LogRT"
colnames(bysubjSem)[3]<-"LogRT"

## main effects:

summary_ttest(
  t.test(subset(bysubjN2,N2Factor=="N2pl")$LogRT,
         subset(bysubjN2,N2Factor=="N2sg")$LogRT,
         paired=TRUE))
```

```
## [1] "t(109)=1.24 p=0.218"
## [1] "est.: 0.02 [-0.01,0.04] ms"
```

```
summary_ttest(
  t.test(subset(bysubjSem,SemFactor=="SemSim")$LogRT,
         subset(bysubjSem,SemFactor=="SemDissim")$LogRT,
         paired=TRUE))
```

```
## [1] "t(109)=2.32 p=0.022"
## [1] "est.: 0.03 [0,0.05] ms"
```

```
## interaction:
## difference between N2pl and N2sg in SemSim level:
d1<- subset(bysubj,N2Factor=="N2pl" & SemFactor=="SemSim")$LogRT-
  subset(bysubj,N2Factor=="N2sg" & SemFactor=="SemSim")$LogRT

## difference between N2pl and N2sg in SemDissim level:
d2<- subset(bysubj,N2Factor=="N2pl" & SemFactor=="SemDissim")$LogRT-
  subset(bysubj,N2Factor=="N2sg" & SemFactor=="SemDissim")$LogRT

## interaction: difference of differences:
summary_ttest(t.test(d2-d1))
```

```
## [1] "t(109)=-0.65 p=0.515"
## [1] "est.: -0.02 [-0.08,0.04] ms"
```

These are essentially equivalent to a single linear mixed model on the aggregated data:

```
bysubj$N2<-ifelse(bysubj$N2Factor=="N2sg",1/2,-1/2)
bysubj$Sem<-ifelse(bysubj$SemFactor=="SemSim",1/2,-1/2)
bysubj$N2xSem<-bysubj$N2*bysubj$Sem
```

```r
m_aggregated<-lmer(LogRT~N2+Sem+N2xSem+(1|Participant),bysubj)
summary(m_aggregated)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: LogRT ~ N2 + Sem + N2xSem + (1 | Participant)
##    Data: bysubj
##
## REML criterion at convergence: -149
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.0635 -0.5117 -0.0312  0.4772  4.3139
##
## Random effects:
##  Groups      Name        Variance Std.Dev.
##  Participant (Intercept) 0.05403  0.2324
##  Residual                0.02196  0.1482
## Number of obs: 440, groups:  Participant, 110
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  6.03697    0.02326 259.539
## N2          -0.01369    0.01413  -0.969
## Sem          0.02788    0.01413   1.974
## N2xSem      -0.02068    0.02826  -0.732
##
## Correlation of Fixed Effects:
##         (Intr) N2    Sem
## N2      0.000
## Sem     0.000  0.000
## N2xSem 0.000  0.000 0.000
```

Some important points to note

- The linear mixed model above on the aggregated data is carrying out all three t-tests (the two main effects and interaction) simultaneously.
- We can only have by participant intercepts, no slopes, as we have only one data point per condition per participant.

```
## This will not work:
#m_aggregatedFULL<-lmer(LogRT~N2+Sem+N2xSem+(1+N2+Sem+N2xSem|Participant),b
```

- Compare with the model with the unaggregated data (this is a more realistic model):

```
summary(m)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log(RT) ~ N2 + Sem + N2xSem + (1 + N2 + Sem + N2xSem | Participan
##     (1 + N2 + Sem + N2xSem | StimSet)
##    Data: df_smithE1
##
## REML criterion at convergence: 2989.1
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.3578 -0.5803 -0.1202  0.4073  8.3840
##
## Random effects:
##  Groups      Name        Variance  Std.Dev. Corr
##  Participant (Intercept) 0.0563127 0.23730
##              N2          0.0025518 0.05052  -0.13
##              Sem         0.0010239 0.03200   0.15 -0.66
##              N2xSem      0.0312132 0.17667  -0.23  0.55 -0.99
##  StimSet     (Intercept) 0.0049071 0.07005
##              N2          0.0005055 0.02248   0.21
##              Sem         0.0006671 0.02583   0.22  1.00
##              N2xSem      0.0049601 0.07043  -0.99 -0.31 -0.32
##  Residual                0.1226560 0.35022
## Number of obs: 3441, groups:  Participant, 110; StimSet, 36
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  6.03753    0.02617 230.702
## N2          -0.01765    0.01348  -1.310
## Sem          0.02927    0.01312   2.230
## N2xSem      -0.02596    0.03163  -0.821
##
## Correlation of Fixed Effects:
##        (Intr) N2     Sem
```

```
## N2     -0.015
## Sem     0.069  0.046
## N2xSem -0.266  0.091 -0.167
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

- Notice that the sign of the main effect or interaction depends on how one sets up the contrast coding!
- Repeated measures ANOVA is equivalent to the three t-tests above:

```
library(rstatix)
```

```
##
## Attaching package: 'rstatix'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
subj_anova<-anova_test(data = bysubj,
           dv = LogRT,
wid = Participant,
within = c(N2Factor,SemFactor)
)

get_anova_table(subj_anova)
```

```
## ANOVA Table (type III tests)
##
##                Effect DFn DFd     F     p p<.05      ges
## 1           N2Factor   1 109 1.035 0.311       0.000622
## 2          SemFactor   1 109 4.644 0.033     * 0.003000
## 3 N2Factor:SemFactor   1 109 0.427 0.515       0.000355
```

Notice that the square roots of the F-scores will be essentially the same (modulo some discrepancies due to numerical inaccuracies) as the t-test values in the paired t-tests:

```
## main effect N2Factor, F-score to t-value of 1.24:
sqrt(1.035)
```

```
## [1] 1.017349
```

```r
## main effect SemFactor, F-score to t-value of 2.32:
sqrt(4.644)
```

```
## [1] 2.154994
```

```r
## interaction, F-score to absolute t-value of 0.65:
sqrt(0.427)
```

```
## [1] 0.6534524
```