Working with Priors in the `brms` package

Bayesian Data Analysis

Steve Buyske

Default priors in `brms`

- Priors in `brms` are more flexible than in `rstanarm`, which makes them a bit more complicated to code. Before we get to the detail
- Overall parameters in the `brms` package have flat priors as defaults.
    - If you read the documentation, you'll see that overall parameters are referred to as *population-level* parameters, which is a better name than *overall*.
- The standard deviation σ has a "half" Student's t prior with 3 degrees of freedom, meaning the positive part of the Student's t distribution.
- Group-level parameters have similar (but not exactly the same) defaults as in `rstanarm`.

Classes of priors

- Unlike `rstanarm`, which used arguments of `prior_intercept`, `prior_aux`, `prior`, and `prior_covariance`,
- for the `brms` package there is a single `prior` argument, will typically be a vector specifying the different types of priors.
- There is an additional function `set_prior()`, that does some of the work behind the scenes.
- The `set_prior()` function has an important argument, `class`, that spells out the type of prior.
- Let's go to an example . . .

Example

- Consider the sleep study example, but now fit using the `brm()` function.

```
sleep2_brm <- brm(Reaction ~ Days + (Days | Subject),  data = sleepstudy, iter = 7500,
                  prior = set_prior("normal(0, 49)", class = "b"))
```

- `class = "b"` refers to the population-level parameters, so this prior, normal(0, 49), applies only to population-level regression parameters (b as in beta).
- If you want to specify different priors for different such parameters, you can add a `coef` argument:
  - `prior = set_prior("normal(0, 49)", class = "b", coef = "Days")`

Example cont.

- To set a prior for the intercept, use `class = "Intercept"`.
- As with `rstanarm`, this is the centered intercept, namely the expected response value when all predictors are at their means.
- So, to set the same prior as before for the regression coefficient of `Days` while adding a prior for the intercept, we might write

```
sleep2_brm_b <-
  brm(
    Reaction ~ Days + (Days | Subject),
    data = sleepstudy,
    iter = 7500,
    prior = c(
      set_prior("normal(0, 49)", class = "b"),
      set_prior("normal(299, 141)", class = "Intercept")
    )
  )
```

`brms` prior for the standard deviation

- For the σ, the standard deviation of observations about their means, the class is “sigma”.

```
sleep2_brm_c <-
  brm(
    Reaction ~ Days + (Days | Subject),
    data = sleepstudy,
    iter = 7500,
    prior = c(
      set_prior("normal(0, 49)", class = "b"),
      set_prior("normal(299, 141)", class = "Intercept"),
      set_prior("exponential(1)", class = "sigma")
    )
  )
```

`brms` prior for the covariance of group-level effects

- The `brms` package splits out the variance and the correlation for group-level effects.
- To change the prior on the variances, use the `class = "sd"` but add a `group` argument:
  - `set_prior("exponential(0.5)", class = "sd", group = "Subject")`
- To change the prior on the correlation, use `class = "cor"`
  - There is only one choice, with one parameter, for the prior, `lkj()`
    - the default is `lkj(1)`, which gives a uniform prior over all correlation matrices
    - A larger value than 1 will make more extreme correlations less likely.
- I sometimes change the prior on the correlation, but almost never change the variance for group-level effects.

Help!

- Except for the priors on the population-level parameters, I usually use the default priors when I use the `brm()` function.
- It's a little annoying to have to change the priors on the population-level parameters from the default of flat priors to weakly informative priors
- There are three approaches you can take:
  - Calculate the priors yourself.
    - Use the mean of the outcomes for the location of the prior for the intercept, and 2.5 (say) times the standard deviation of the outcomes for the scale.
    - Use 0 for the location of the priors for the regression coefficients, and 2.5 (say) times sd(y)/sd(x) for the scale.
  - Run `stan_glm()` or `stan_glmer()` with `iter = 1`, ignore the errors, and use `describe_prior()` or `prior_summary()` to see what priors the `rstanarm` would have used.
  - Or …

Help! cont

- There is actual a function `auto_prior()` in the `sjstats` package that will create similar priors to `rstanarm`
- **Don't** include the group-level part in the formula

```
sjstats::auto_prior(formula = Reaction ~ Days, data = sleepstudy, gaussian = TRUE)
```

```
##              prior     class coef group resp dpar nlpar bound source
##  normal(0, 563.29) Intercept                                     user
##   normal(0, 48.89)         b Days                                user
```

- Two notes:
  - Much to my annoyance, the prior for the intercept is a bit different than `rstanarm`.
    - It gives a location of 0 and a scaled standard deviation of 10, not 2.5

- The `auto_prior()` function is the only thing I want from the `sjstats`, so instead of using `library(sjstats)` early on and then `auto_prior()` here, I use `sjstats::auto_prior()` to pluck out the function without loading the package.

The output of `prior_summary()`

- The output of `prior_summary()` on the result of a `brm()` fit is not designed for beginners. Here's a run through.