EZ-COVID data: First steps

Adriana F. Chávez, Eunice Shin and Joachim Vandekerckhove

2024-12-04

Goal:

We'll get familiar with the general procedure to fit the Bayesian Hierarchical EZ-DDM to real data. In this first approximation, we'll do a simple analysis where we'll focus on the data collected across participants. We won't be using any demographic information yet.

For this first step, we are going to be working mostly with file `./ez-covid/data/dataForAnalysisMergedN68.csv`.

Step 0: Get familiar with the data

Let's take a look at file `dataForAnalysisMergedN68.csv` located in the `./ez-covid/data` folder.

```
data <- read.csv("./ez-covid/data/dataForAnalysisMergedN68.csv")
head(data,3)
```

```
##   X participant_id age race ethnicity sex education covid raceNew schedule day
## 1 1           4000  36    0         0   0         7     0       0        1   7
## 2 2           4000  36    0         0   0         7     0       0        1   7
## 3 3           4000  36    0         0   0         7     0       0        1   7
##          version sequential_task_order task_time trial_num trial_type
## 1 SP_ST+_RT+_PC+                 task2       4.8         1          1
## 2 SP_ST+_RT+_PC+                 task2       4.8         2          1
## 3 SP_ST+_RT+_PC+                 task2       4.8         3          0
##   button_pressed response_time HIT FA MISS CR  pack_id accuracy trial_type_text
## 1              1          1814   1  0    0  0 pack_1_7        1       different
## 2              1          1134   1  0    0  0 pack_1_7        1       different
## 3              0           920   0  0    0  1 pack_1_7        1            same
##   button_pressed_text       versionF versionCode StudyTime StudyTimeCoded
## 1           different SP_ST+_RT+_PC+           8       ST+              1
## 2           different SP_ST+_RT+_PC+           8       ST+              1
## 3                same SP_ST+_RT+_PC+           8       ST+              1
##   ChoiceUrgency ChoiceUrgencyCoded ProbOfChange ProbOfChangeCoded ProbeType
## 1           RT+                  0          PC+                 1        SP
## 2           RT+                  0          PC+                 1        SP
## 3           RT+                  0          PC+                 1        SP
##   ProbeTypeCoded
## 1              1
## 2              1
## 3              1
```

```r
ncol(data) # No. of columns
```

```
## [1] 36
```

```r
nrow(data) # No of rows
```

```
## [1] 58978
```

The dataset contains a total of 58,978 rows. Each row corresponds to a single trial observed in the experiment.

There are 36 columns in our dataset. This means that for each trial, we have information about 36 different variables. For this first exercise, we are only going to focus on the following columns:

- "participant_id" (column `2`): For each trial observed (row), this column indicates the ID of the participant it belongs to.
- "response_time" (column `18`): For each trial (row), we store the response time (i.e., the time elapsed since the beginning of the trial and the moment the participant makes a response)
- accuracy (column `24`): For each trial (row), we identify whether the response made by the participant was correct (`accuracy = 1`) or not (`accuracy = 0`).

Step 1: Clean the data

1. How many trials do we have per participant? Do we have the same number of trials per participant? **Make a histogram for the number of trials per participants**.

You should see there seems to be 1 participant that was fewer trials than anybody else in the experiment. **What is the ID of this participant?** Once you have identified them **Remove all trials from this participant from the dataset**.

2. Now, let's look at the distribution of response times. **Make a histogram for all the response times observed in the experiment**.

You should find that most response times are smaller than 4500ms, but there's a few outliers. **Remove trials with a RT larger than 4500ms in the dataset**.

3. Finally, let's look at how well participants did in the task. For this, we'll focus on the **accuracy rate** (that is, the proportion of correct responses each participant made during the experiment). **Compute the accuracy rates per participant, and show their distribution in a histogram**.

You should see that most participants had an accuracy rate of at least 70%. There's one participant who is the exception. **What is the ID for the participant with the lowest accuracy rate?** Once you have identified them, **remove all trials belonging to this participant from the dataset**.

Step 2: Computing EZ summary statistics

1. Compute the three summary statistics used by the EZ-DDM **by participant**:
   - **Compute the Accuracy rate per participant**
   - **Compute the Mean RT per participant**
   - **Compute the Variance RT per participant**
2. We will also need to keep track of the number of trials observed for each participant to compute the variances of the generative distributions.
   - **Compute the number of trials per participant**
3. We will need this to run our JAGS model: How many participants are there? **Create a variable storing the number of participants in the experiment**.

Step 2: Prepare JAGS!

1. Write the JAGS model (We can use the same one as the one we used for our last assignment)

Note: This time we have no way of knowing which would be the "true parameter values" generating the data observed. We may need to change the priors we use for `drift_mean`, `drift_sdev`, `bound_mean`, `bound_sdev` and `nondt_mean`, `nondt_sdev`. Keep this in mind when running JAGS and exploring the posterior distributions requested.

2. Create a `parameters` object that specifies which parameters we want to track.
3. Define `n.chains`, `n.iter`, `n.burnin` and `n.thin`.
4. Create a `data_toJAGS` object. This must be a list pairing the summary statistics you computed on the last step, to the variables used or referred in the JAGS model.
5. Create a `myinits` object.

Step 4: RUN JAGS

Run your JAGS model using the `jags()` function! Make sure to load all necessary information and settings.

Results

1. Plot the posterior distributions for `drift_mean`, `nondt_mean` and `bound_mean`.