GSandow_Simulation_DDM

Greta Sandow

2024-06-11

Reinforcement Learning and Decision Making: Computational and Neural Mechanisms

Drift Diffusion Model (DDM) - Data Simulation

The DDM can be used to simulate artificial data by adjusting the free parameters between groups based on a review of the literature. This data could be used for further analysis, but it was not included in my final assignment.

As the processes of the DDM are inspired by brain mechanisms of decision-making, we can vary the free parameters between the participants' measurement time points to generate human-like data. If we can create a model that produces data resembling human behavior under different conditions, we might have identified the underlying process that generates such data.

To start, let us load the necessary packages.

```
setwd("~/Documents/Cognitive Neuroscience/Reinforcement Learning & Decision Making")
library(tidyverse)
library(readr)
library(ggplot2)
library(papaja)
```

Creating the Drift Diffusion Model Function

Before I adjust any of the parameters, I will create the base function for the DDM. This function takes the arguments for the decision boundary/threshold, the evidence for the two decision options (IA and IB), and the standard deviation of the Gaussian noise. Initially, the non-decision time and bias are determined from normal distributions. The current time point is started at 0, to which 1 will be added for each time the while loop runs. This loop will continue running until the accumulator has reached one of the two decision boundaries, at which point the function saves the output of the simulated response times and the chosen decision option. This information is necessary for any further analyses. This implementation was done according to the DDM Equation, in which at stands for the evidence accumulation process a at time point t. The drift rate (IA - IB) is represented by the evidence for one decision B (incorrect decision) subtracted from the evidence for the other decision A (correct decision) and the Gaussian noise N with mean 0 and a specified standard deviation.

$$\mathrm{at} = \mathrm{at} - 1 + (\mathrm{IA} - \mathrm{IB}) + \mathrm{N}(0, \sigma)$$

I will change the drift rate (through altered evidence) and the decision boundaries between the conditions.

```
ddm <- function(threshold = 10, I1 = 0.2, I2 = 0.1, SD = 1) { # threshold, evidence 1
& 2, SD of noise
  t <- 0 # set time point to 0
  time_delay <- rnorm(1, mean = 400, sd = 20) # non-decision time
  bias <- rnorm(1, mean = 0.5, sd = 0.01) # response bias
  accum <- threshold * bias # accumulator
  drift <- I1 - I2 # drift rate
  done <- FALSE

  while (done == FALSE) {
    accum <- accum + drift + rnorm(1, mean = 0, sd = SD) # evidence accumulation
    t <- t + 1 # adding a time step

    if (accum >= threshold) { # decision 1
      results <- list("time" = t + time_delay, "decision" = "decision_1")
      done <- TRUE
    } else if (accum <= 0) { # decision 2
      results <- list("time" = t + time_delay, "decision" = "decision_2")
      done <- TRUE
    }
  }

  return(results)
}
```

Simulating the Data

I will generate a data set with the response times and accuracy of 15 participants measured at two time points with 500 trials each.

```
# Participants and Trials
n_participants <- 15
n_trials <- 500
```

In my data set, I want the participants to have different parameter values depending on the condition (measured in childhood and adolescence). For this, I make a function that generates the parameter values for each condition from a normal distribution.

```
# Generating the Parameters
parameters <- function() {
  list(
    child = list( # Children: Normal Distributions for Parameter Generation
      threshold = rnorm(1, mean = 30, sd = 4),
      I1 = rnorm(1, mean = 0.2, sd = 0.05),   # Drift rate = 0.05
      I2 = rnorm(1, mean = 0.1, sd = 0.05),
      SD = rnorm(1, mean = 1, sd = 0.05)
    ),
    adolescent = list( # Adolescents: Normal Distributions for Parameter Generation
      threshold = rnorm(1, mean = 20, sd = 4),
      I1 = rnorm(1, mean = 0.2, sd = 0.05),    # Drift rate = 0.1
      I2 = rnorm(1, mean = 0.05, sd = 0.05),
      SD = rnorm(1, mean = 1, sd = 0.05)
    )
  )
}
```

Next, I need a function that combines these parameters in the DDM for a single participant.

```
# Function to simulate data for a single participant #
participant_sim <- function(ID, time_point, parameters) {
  results <- vector("list", n_trials)  # list to store the results of each trial simu
lation

  for (trial in 1:n_trials) {
    results[[trial]] <- ddm(threshold = parameters$threshold,
                            I1 = parameters$I1,
                            I2 = parameters$I2,
                            SD = parameters$SD)
  }

  data <- tibble( # create data frame
    trial = 1:n_trials # trial column  from 1 to n_trials
  ) %>%
    bind_cols(bind_rows(results)) %>% # combine the list of trial results into a data
frame
    mutate(participant = ID,
           time_point = time_point,
           threshold = parameters$threshold,
           evidence_1 = parameters$I1,
           evidence_2 = parameters$I2,
           SD = parameters$SD)

  return(data)
}
```

Finally, I can use this participant function and loop it as many times as needed to create the final data set.

```
# Generate data for each participant at each time point

simulated_ddm_data <- data.frame() # start an empty data frame

for (p in 1:n_participants) { # loop over participants
  participant_parameters <- parameters() # parameters using earlier function

  # loop over time points (child, adolescent)
  for (time_point in names(participant_parameters)) {
    # simulate data for each participant at each time point
    participant_data <- participant_sim(p, time_point, participant_parameters[[time_p
oint]])
    # combine data using rbind
    simulated_ddm_data <- rbind(simulated_ddm_data, participant_data)
  }
}
```

I will reorganize the data, just so it is easier for me to handle.

```
# Reorganizing the data set
simulated_ddm_data <- relocate(simulated_ddm_data, participant, .before = trial)
simulated_ddm_data <- relocate(simulated_ddm_data, time_point, .before = trial)
simulated_ddm_data <- mutate(simulated_ddm_data, drift_rate = evidence_1 - evidence_
2)
```

As a final step, the data needs to be saved in case it is needed again later!

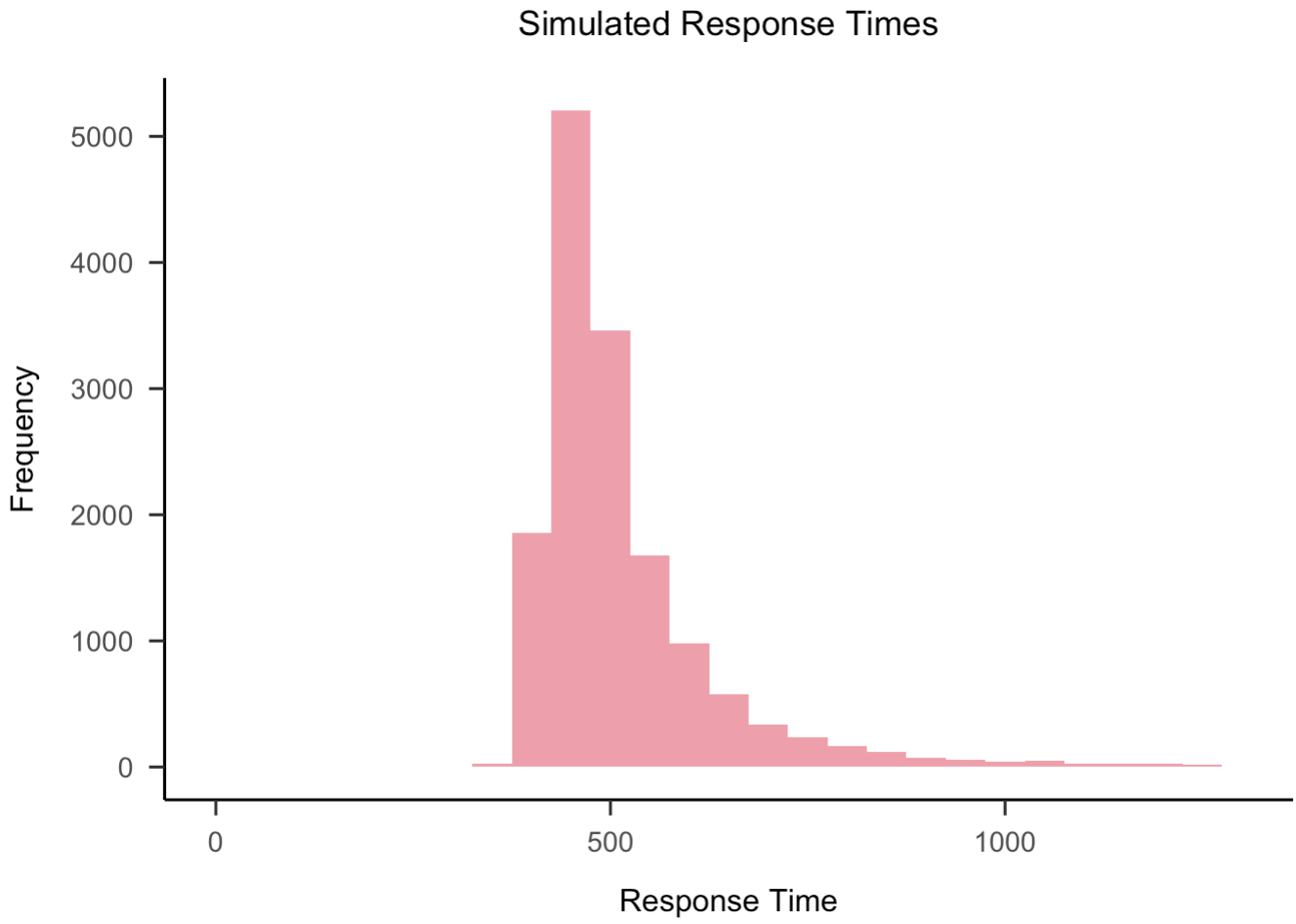
```
# Save as CSV file
write_csv(simulated_ddm_data, "simulated_ddm_data.csv")
head(simulated_ddm_data, n=10)
```

```
## # A tibble: 10 × 10
##    participant time_point trial  time decision   threshold evidence_1 evidence_2
##          <int> <chr>      <int> <dbl> <chr>          <dbl>      <dbl>      <dbl>
##  1           1 child          1  471. decision_1      34.6      0.279     0.0685
##  2           1 child          2  427. decision_1      34.6      0.279     0.0685
##  3           1 child          3  421. decision_1      34.6      0.279     0.0685
##  4           1 child          4  463. decision_1      34.6      0.279     0.0685
##  5           1 child          5  476. decision_1      34.6      0.279     0.0685
##  6           1 child          6  384. decision_1      34.6      0.279     0.0685
##  7           1 child          7  502. decision_1      34.6      0.279     0.0685
##  8           1 child          8  428. decision_1      34.6      0.279     0.0685
##  9           1 child          9  428. decision_1      34.6      0.279     0.0685
## 10           1 child         10  467. decision_1      34.6      0.279     0.0685
## # ℹ 2 more variables: SD <dbl>, drift_rate <dbl>
```

Descriptives and Visualizations

Let us take a look at the data we just generated.

```
# Response Time Distribution
ggplot(simulated_ddm_data, aes(x = time)) +
  geom_histogram(binwidth = 50, fill = "lightpink2") +
  xlim(c(0,1300))+
  labs(title = "Simulated Response Times",
       x = "Response Time",
       y = "Frequency")+
  theme_apa()
```



I will also look at the response time and accuracy in general and between the two conditions.

```
# Seperate Children and Adolescents
child_data <- filter(simulated_ddm_data, time_point == "child")
adolescent_data <- filter(simulated_ddm_data, time_point == "adolescent")

# Response times
print(paste("Minimum Response Time:", round(min(simulated_ddm_data$time), digits =
2)))
```

```
## [1] "Minimum Response Time: 350.93"
```

```
print(paste("Maximum Response Time:", round(max(simulated_ddm_data$time), digits =
2)))
```

```
## [1] "Maximum Response Time: 2216.91"
```

```
print(paste("Average Response Time:", round(mean(simulated_ddm_data$time), digits =
2)))
```

```
## [1] "Average Response Time: 516.88"
```

```
print(paste("SD Response Time:", round(sd(simulated_ddm_data$time), digits = 2)))
```

```
## [1] "SD Response Time: 130.23"
```

```
print(paste("Median Response Time:", round(median(simulated_ddm_data$time), digits =
2)))
```

```
## [1] "Median Response Time: 479.27"
```

```
print(paste("Median Response Time Children:", round(median(child_data$time), digits =
2)))
```

```
## [1] "Median Response Time Children: 511.19"
```

```
print(paste("Median Response Time Adolescents:", round(median(adolescent_data$time),
digits = 2)))
```

```
## [1] "Median Response Time Adolescents: 457.26"
```

```
# Accuracy
decision_counts <- table(simulated_ddm_data$decision)
correct_decisions <- decision_counts["decision_1"]
accuracy <-correct_decisions / sum(decision_counts)*100
print(paste("Overall Accuracy:", round(accuracy, digits = 2), "%"))
```

```
## [1] "Overall Accuracy: 85.9 %"
```

```
decision_counts_child <- table(child_data$decision)
correct_decisions_child <- decision_counts_child["decision_1"]
accuracy_child <-correct_decisions_child / sum(decision_counts_child)*100
print(paste("Accuracy Children:", round(accuracy_child, digits = 2), "%"))
```

```
## [1] "Accuracy Children: 80.03 %"
```

```
decision_counts_adol <- table(adolescent_data$decision)
correct_decisions_adol <- decision_counts_adol["decision_1"]
accuracy_adol <-correct_decisions_adol / sum(decision_counts_adol)*100
print(paste("Accuracy Adolescents:", round(accuracy_adol, digits = 2), "%"))
```

```
## [1] "Accuracy Adolescents: 91.77 %"
```

This marks the end of the data simulation code.