My_Agent - R Markdown

In this document, we will first display our code. Next, we will explain the strategy that we used for this tournament.

```
library(R6)

Agent <- R6Class(
  "Agent",

  public = list(
    bid = NULL,
    book = NULL,
    greeting = NULL,
    id = NULL,
    opponent_id = NULL,
    opponent_greeting = NULL,
    round = NULL,
    response = NULL,
    my_move=NULL,
    opponent_move=NULL,

    set_book = function(book = NA) {
      self$book <- book
    },

    set_id = function(id = NA) {
      self$id = id
    },

    set_opponent_id = function(opponent_id = NA) {
      self$opponent_id = opponent_id
    },

    set_response = function(response = NA) {
      self$response <- response
    },

    set_round = function(round = NA) {
      self$round <- round
    },

    set_greeting = function() {
      self$greeting <- "Hi!"
    },

    receive_greeting = function(greeting = NA) {
      self$opponent_greeting = greeting
    },

    latest_my_move = function(my_move = NA) {
      my_index <- which(self$book$id1 == self$id | self$book$id2 == self$id)
      latest_index <- max(my_index)
      latest_round <- self$book[latest_index,]
      if (is.data.frame(latest_round) && nrow(latest_round)==0){
          my_move <- "cooperate"
      }
      else {
        if (latest_round$id1 == self$id) {
          my_move <- latest_round$bid1
        }
        else {
```

```
          my_move <- latest_round$bid2
        }
      }
      return(my_move)
    },

    latest_opponent_move = function(opponent_move=NA) {
      my_index <- which(self$book$id1 == self$id | self$book$id2 == self$id)
      latest_index <- max(my_index)
      latest_round <- self$book[latest_index,]
      if (is.data.frame(latest_round) && nrow(latest_round)==0){
        opponent_move <- "cooperate"
      }
      else {
        if (latest_round$id1 == self$id) {
          opponent_move <- latest_round$bid2
        }
        else {
          opponent_move <- latest_round$bid1
        }
      }
      return(opponent_move)
    },

    get_bid = function() {
      my_index <- which(self$book$id1 == self$id | self$book$id2 == self$id)

      if (length(my_index)<2) {
        self$bid <- "cooperate"
      }
      else if (self$latest_my_move() == "cooperate"){
        if (self$latest_opponent_move() == "cooperate"){
          self$bid <- "cooperate"
        }
        else {
          self$bid <- "defect"
        }
      }
      else {
        self$bid <- "defect"
      }
    },
    formulate_bid = function() {
      self$get_bid()
    }

  )
)
```

The strategy we have chosen is called “spiteful_cc”" which was introduced and analyzed in the paper of Mathieu and Delahaye (2017). Basically, our agent will cooperate for the first 2 rounds and then continue cooperating until an opponent defects. From that moment onwards, our agent will always defect.

We designed our code to execute the following tasks: First, our agent will cooperate in the first two turns no matter what our opponent does. After that, our agent will decide its move based on whether our agent was betrayed or not (in other words, what the opponent did to us) in the previous round .

Besides the existing funcitons in the example agent, we have added a few functions of our own.

1. The “latest_my_move” function aims to find the last move of our agent by scanning the table (specifically columns id1 and id2) for the id of our agent and the corresponding move for that id.
2. The “latest_opponent_move”" is similar to the latest_my_move function but this is used to find the opponent’s last move instead.
3. Finally is the “get_bid” function. We set our agent to cooperate in the first two rounds it plays in the tournament. Next, our agent cooperates as long as the opponent’s agent cooperates but once an opponent defects, our agent defects thereafter.

Reference: Mathieu, P., & Delahaye, J.-P. (2017). New Winning Strategies for the Iterated Prisoners Dilemma. Journal of Artificial Societies and Social Simulation, 20(4). doi: 10.18564/jasss.3517