Overview of the Functions

Raphaël Morsomme

December 11, 2018

```
clean_pop_B <- function(pop, cards){
  pop[max(cards),    ,] <- "Call"
  pop[          , "0",] <- "Call"
  return(pop)
}
```

```
confront <- function(strategy_A, strategy_B, n_card, dim_mat, bets, ante, win_game){

  bet_A    <- array(rep(strategy_A, each = n_card       ), dim = dim_mat)
  action_B <- array(strategy_B[ , match(strategy_A, bets)], dim = dim_mat)
  gain_A                   <- (ante + bet_A) * win_game
  gain_A[action_B=="Fold"] <-  ante

  profitability_A <- mean(gain_A)

  return(profitability_A)

}
```

```
confront_populations <- function(pop_A, pop_B, name_strategy, fitness,
                                 n_card, dim_mat, bets, ante, win_game){

  for(strat_a in name_strategy){
    for(strat_b in name_strategy){
      fitness[strat_b, strat_a] <- confront(strategy_A = pop_A[ , strat_a  ],
                                            strategy_B = pop_B[ , , strat_b],
                                            n_card = n_card, dim_mat = dim_mat,
                                            bets = bets, ante = ante,
                                            win_game = win_game)
    } # end-for
  } # end-for
  return(fitness)
}
```

```
generate_A <- function(fitness, pop, n_parents, dim_pop_A, dimname_pop_A,
                       n_strategy, bets, mutation_rate){

  # Parent Selection
  fitness_strategy <- colMeans(fitness)
  fitness_parents  <- head(sort(fitness_strategy, decreasing  = T), n_parents)
  name_parents     <- names(fitness_parents)
  parents          <- pop[ , name_parents]

  # Child Generation
  pop <- array(sample(name_parents, size = prod(dim_pop_A), replace = T,
                      prob = exp(fitness_parents)),
               dim = dim_pop_A, dimnames = dimname_pop_A)
```

```
  for(parent in name_parents){

    strategy_parent <- parents[ , parent]
    position_parent <- pop == parent
    pop[position_parent] <- rep(strategy_parent, n_strategy)[position_parent]

  }

  pop <- array(as.numeric(pop), dim = dim_pop_A, dimnames = dimname_pop_A)

  # Mutation
  mutation_position <- array(sample(c(T, F), size = prod(dim_pop_A), T,
                                    prob = c(mutation_rate, 1-mutation_rate)),
                             dim = dim_pop_A, dimnames = dimname_pop_A)

  n_mutations      <- sum(mutation_position)
  mutation_outcome <- sample(bets, size = n_mutations, replace = T)

  pop[mutation_position] <- mutation_outcome

  return(pop)

}
```

```
generate_B <- function(fitness, pop, n_parents, dim_pop_B, dimname_pop_B,
                       n_strategy, mutation_rate, cards){

  # Parent Selection
  fitness          <- - fitness
  fitness_strategy <- rowMeans(fitness)
  fitness_parents  <- head(sort(fitness_strategy, decreasing  = T), n_parents)
  name_parents     <- names(fitness_parents)
  parents          <- pop[ , , name_parents]

  # Child Generation
  pop <- array(sample(name_parents, size = prod(dim_pop_B), replace = T,
                      prob = exp(fitness_parents)),
               dim = dim_pop_B, dimnames = dimname_pop_B)

  for(parent in name_parents){

    strategy_parent <- parents[ , , parent]
    position_parent <- pop == parent

    pop[position_parent] <- rep(strategy_parent, n_strategy)[position_parent]

  }

  # Mutation
  mutation_rate <- 2 * mutation_rate
  # double mutation_rate because since player B has only two possible
  # actions i.e. "Call" or "Fold", half of the mutations will have no effect.
  mutation_position <- array(sample(c(T, F), size = prod(dim_pop_B), replace = T,
                                    prob = c(mutation_rate, 1-mutation_rate)),
```

```
                          dim = dim_pop_B, dimnames = dimname_pop_B)

  n_mutations            <- sum(mutation_position)
  mutation_outcome       <- sample(c("Call", "Fold"), n_mutations, T)
  pop[mutation_position] <- mutation_outcome

  pop <- clean_pop_B(pop = pop, cards = cards)

  return(pop)

}
```

```
my_genetic_algorithm <- function(cards = 1:10, bets = seq(0,20,2), ante = 5,
                                 n_strategy = 200, n_generations = 200,
                                 prop_parents = 2/3, mutation_rate = 0.05,
                                 gen_print = seq(0, n_generations, by = 40)){

  # Setup
  n_card        <- length(cards)
  n_bet         <- length(bets)
  n_parents     <- n_strategy * prop_parents

  name_strategy <- paste("s", 1:n_strategy, sep="")

  dim_pop_A     <- c(n_card, n_strategy)
  dim_pop_B     <- c(n_card, n_bet, n_strategy)
  dim_mat       <- c(n_card, n_card)
  dim_fit       <- c(n_strategy, n_strategy)

  dimname_pop_A <- list(cards, name_strategy)
  dimname_pop_B <- list(cards, bets, name_strategy)
  dimname_mat   <- list(cards, cards)
  dimname_fit   <- list(name_strategy, name_strategy)

  win_game      <- array(numeric(n_card*n_card), dim = dim_mat)
  upperTriangle(win_game, diag = F) <-  1
  lowerTriangle(win_game, diag = F) <- -1

  fitness <- array(NA, dim = dim_fit, dimnames = dimname_fit)

  par(mfrow=c(2,2), cex = 1.1)

  # Initialization
  pop_A <- array(sample(bets, size = prod(dim_pop_A), replace = T),
                 dim = dim_pop_A, dimnames = dimname_pop_A)
  pop_B <- array(sample(c("Call", "Fold"), size = prod(dim_pop_B), replace = T),
                 dim = dim_pop_B, dimnames = dimname_pop_B)
  pop_B <- clean_pop_B(pop_B, cards = cards)

  generation <- 0
  gain_A     <- 0
  call_B     <- 0.5

  # Loop
```

```
  while(generation <= n_generations){

    # Plots
    if(generation %in% gen_print){

      barplot(rowMeans(pop_A), ylim = c(0, max(bets)),
              main = "Average Strategy for Player A",
              xlab = "Player A's Card", ylab = "Average Bet")
      abline(h=bets, lty=2)

      mtext(paste("Generation", generation), cex = 2, adj = -0.1, line = 2.5)

      image(cards, bets, rowMeans(pop_B == "Call", dims = 2),
            col = gray((100 : 0) / 100),
            main = "Average Strategy for Player B",
            xlab = "Player B's Card", ylab = "Average Action")

      plot(0 : generation, gain_A, type = "l", xlim = c(0, generation),
           main = "Average Gain/Loss (Player A)",
           xlab = "Generation", ylab = "Player A's Average Gain/Loss")
      abline(h = 0, lty = 0)
      abline(h = seq(-10, 10, 0.5), lty = 2)

      plot(0 : generation, call_B, type = "l", xlim = c(0, generation),
           main = "Proportion of Calls (Player B)",
           xlab = "Generation", ylab = "Average Proportion of Calls")
      abline(h = 0, lty = 0)
      abline(h = seq(-10, 10, 0.04), lty = 2)

    } # close if-statement

    fitness <- confront_populations(pop_A = pop_A, pop_B = pop_B,
                                    name_strategy = name_strategy,
                                    fitness = fitness, n_card = n_card,
                                    dim_mat = dim_mat, bets = bets,
                                    ante = ante, win_game = win_game)

    pop_A <- generate_A(fitness = fitness, pop = pop_A, n_parents = n_parents,
                        dim_pop_A = dim_pop_A, dimname_pop_A = dimname_pop_A,
                        n_strategy = n_strategy, bets = bets,
                        mutation_rate = mutation_rate)

    pop_B <- generate_B(fitness = fitness, pop = pop_B, n_parents = n_parents,
                        dim_pop_B = dim_pop_B, dimname_pop_B = dimname_pop_B,
                        n_strategy = n_strategy, cards = cards,
                        mutation_rate = mutation_rate)

    gain_A <- c(gain_A, mean(fitness))
    call_B <- c(call_B, mean(pop_B == "Call"))
    generation <- generation + 1

  } # close for-loop
}
```